FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN ELECTRONICS AND COMMUNICATIONS ENGINEERING

MASTER'S THESIS

Model-based systems engineering approach in phased antenna array design and optimization

Author	Pyry Salonpää
Supervisor	Aarno Pärssinen
Second Examiner	Marko Leinonen
(Technical Advisor	Tero Kangasvieri)

July 2021

Salonpää P. (2021) Model-based systems engineering approach in phased antenna array design and optimization. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Electronics and Communications Engineering. Master's Thesis, 65 p.

ABSTRACT

This thesis introduces the concept of Model-based systems engineering by providing an example on how the hardware aspects of a phased antenna array can be modeled with a system modeling language SysML in a modeling software application Cameo Systems modeler, and demonstrates how the resulting system model is used as a central hub for integration with the analysis of a phased antenna array.

This thesis covers the creation of analysis models at three different fidelity levels for a phased antenna array that operates in the frequency range from 3.4 to 4 gigahertz. The models are created with the electromagnetic modeling and simulation software Ansys HFSS, and the programming and numeric computing platform MATLAB, which is used to create a script that handles post-processing of the simulation results.

The lowest fidelity analysis model is automated in the Multi-disciplinary analysis and optimization tool ModelCenter. The result is an analysis workflow with configurable design parameters as its inputs and performance evaluation parameters as its outputs. The workflow combines and automates the consequent execution of the HFSS electromagnetic model and the post-processing MATLAB script. Afterwards, the workflow is integrated with the system model, which enables the use of requirements in the analysis, and the ability to upload designs achieved with the analysis to the system model.

This connected workflow is used to perform a Design of experiments and a machine learning algorithm driven optimization on the phased antenna array, with the goal of finding the best possible spacings between the individual radiating elements in the array. The Design of experiments produces graphs that visualize statistical relationships between the antenna array's design variables and its performance evaluation parameters.

The optimization produces a graph that visualizes a pareto front between different performance evaluation parameters. In other words, the graph shows the design alternatives that cannot be further improved in any parameter without degrading another.

This graph is used to make an informed decision on the best radiating element spacings in the antenna array. This results in 50 millimetres for the vertical spacing and 40 millimetres for the horizontal spacing in this example. Finally, the design option is uploaded to the system model, which concludes the demonstration of system and analysis modeling and their integrated usage in the design and optimization of a phased antenna array.

Key words: Model-based Systems Engineering, System modeling, SysML, Cameo Systems Modeler, Phased Antenna Arrays, Electromagnetic simulation, Ansys HFSS, MATLAB, Multi-disciplinary Analysis and Optimization, Design of Experiments, Optimization, ModelCenter.

Salonpää P. (2021) Vaiheistettujen antenniryhmien suunnittelu ja optimointi mallipohjaisen järjestelmäsuunnittelun avulla. Oulun yliopisto, tieto- ja sähkötekniikan tiedekunta, elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Diplomityö, 65 p.

TIIVISTELMÄ

Tässä diplomityössä käydään läpi mallipohjaisen järjestelmäsuunnittelun konsepti, sekä osoitetaan esimerkin avulla, kuinka sitä käytetään vaiheistettujen antenniryhmien mallintamiseen SysML-järjestelmänmallinnuskielellä, Cameo Systems Modeler-työkalussa. Tämän lisäksi työssä esitetään, kuinka mallinuksesta syntyvää järjestelmämallia käytetään integroinnin keskuksena vaiheistetun antenniryhmän suunnittelun analysoinnille.

Työssä käydään läpi kolmen eri tarkkuustason mallin luonti vaiheistetulle antenniryhmälle, jonka toimintataajuusalue ulottuu 3.4:stä gigahertsistä neljään gigahertsiin. Mallit luodaan käyttämällä sähkömagneettista mallinnus-ja simulointi ohjelmistoa nimeltään Ansys HFSS, sekä numeerista laskenta-ja ohjelmointialustaa nimeltään MATLAB, jolla luodaan skripti simuloinnin tulosten jälkikäsittelyä varten.

Tämän jälkeen alimman tarkkuustason analyysimalli automatisoidaan monitieteisellä analyysi-ja optimointi työkalulla nimeltään ModelCenter. Tämä tehdään rakentamalla analyysin työnkulku, jonka tulona on antenniryhmän suunnittelumuuttujia ja lähtönä sen suorituskykyä kuvaavia parametreja. Analyysin työnkulku yhdistää sekä automatisoi sähkömagneettisen HFSS-mallin ja MATLAB-jälkikäsittelyskriptin peräkkäisen ajamisen ModelCenterissä. Tämän jälkeen analyysin työnkulku integroidaan järjestelmämallin kanssa. Tämä mahdollistaa vaatimusten käyttämisen analyyseissa sekä kyvyn ladata analyysin perusteella saatuja suunnitteluvaihtoehtoja järjestelmämalliin.

Tätä kytkettyä analyysin työnkulkua käytetään vaiheistetun antenniryhmän kokeelliseen suunnitteluun sekä koneoppimisen algoritmeja käyttävään optimointiin, joiden tavoitteena on löytää parhaat mahdolliset antenniryhmän yksittäisten säteilevien elementtien väliset etäisyydet. Kokeellinen suunnittelu tuottaa kuvaajia, jotka visualisoivat antenniryhmän suunnittelumuuttujien ja suorituskykyä kuvaavien parametrien välisiä tilastollisia riippuvuussuhteita, tehden niiden ymmärtämisestä helppoa.

Optimointi tuottaa kuvaajan, joka visualisoi eri suunnitteluvaihtoehdoilla saatavien suorituskykyä kuvaavien parametrien välistä pareto-tehokkuutta. Toisin sanoen kuvaajasta nähdään parhaat suunnitteluvaihtoehdot, joissa minkään suorituskykyä kuvaavan parametrin arvoa ei voida enää parantaa huonontamatta jonkun toisen arvoa.

Tämän kuvaajan perusteella tehdään päätös parhaista mahdollisista antenniryhmän säteilevien elementtien välisistä etäisyyksistä. Tulos johon esimerkissä päädytään on 50 millimetriä korkeussuunnassa ja 40 millimetriä sivusuunnassa. Lopuksi tämä suunnitteluvaihtoehto ladataan järjestelmämalliin, joka päättää havainnollistavan esimerkin järjestelmä ja analyysimallinnuksesta, sekä niiden yhdistetystä käytöstä vaiheistettujen antenniryhmien suunnittelussa ja optimoinnissa.

Avainsanat: Mallipohjainen järjestelmäsuunnittelu, Järjestelmämallinnus, SysML, Cameo Systems Modeler, Vaiheistetut antenniryhmät, Sähkömagneettinen simulaatio, Ansys HFSS, MATLAB, Monitieteinen analyysi ja optimointi, Kokeellinen suunnittelu, Optimointi, ModelCenter.

TABLE OF CONTENTS

FOREWORD

The target of this thesis is to study and demonstrate how model-based systems engineering methods and software applications can be used to design and optimize phased antenna arrays. This involves modeling the hardware aspects of a phased antenna array with a system modeling language SysML, creating electromagnetic analysis models, integrating these two domains, and finally using them together to design a phased antenna array. This work was done as a part of the VETURI project between Nokia Solutions and Networks Oy and the University of Oulu.

First and foremost, I would like to show my gratitude to my technical advisor Tero Kangasvieri and my supervisor professor Aarno Pärssinen for their guidance and valuable comments. In addition, I want to show my appreciation to Teuvo Virsu who leads the virtual platform project at Nokia under which this thesis was conducted. I also want to give special thanks to my other colleagues at Nokia, Tiina Rantala, Marko Kokko, Mikko Kaarlela, and Tomi Haapala, who all contributed to make this thesis a success. Furthermore, I would like to express my gratitude to Hawal Rashid and Ilya Tolchinsky from Ansys, Inc. who provided assistance with the usage of the software applications used in this thesis. Finally, I would like to express my deepest gratitude to my wife and friends who supported me emotionally and also helped me in the proofreading of this thesis.

Oulu, July 30th, 2021

Pyry Salonpää

LIST OF ABBREVIATIONS AND SYMBOLS

RRH	Remote radio head
RF	Radio frequency
5G	Fifth-generation mobile network technology
MBSE	Model-based systems engineering
GHz	Gigahertz
MDAO	Multi-disciplinary analysis and optimization
INCOSE	International council on systems engineering
IEEE	Institute of electrical and electronics engineers
CAD	Computer-aided design
UML	Unified modeling language
SysML	Systems modeling language
OMG	Object management group
TWC	Teamwork Cloud
AF	Array factor
NGMN	Next generation mobile networks alliance
BASTA	Standards for base station antenna performance validation
dB	Decibel
HPBW	Half-power beamwidth
FBR	Front-to-back ratio
GLS	Grating lobe suppression
SLS	Sidelobe suppression
XPD	Cross-polar discrimination
RL	Return loss
EM	Electromagnetic
GUI	Graphical user interface
3D	Three-dimensional
FEM	Finite element method
DOE	Design of experiments
MIMO	Multiple-input and multiple-output
PCB	Printed circuit board
IP	Intellectual property
BDD	Block definition diagram
IBD	Internal block definition diagram

r	Radial distance
θ	Polar angle
φ	Azimuthal angle
M	Number of vertical elements in an array
dx	Vertical spacing between elements in an array
N	Number of horizontal elements in an array
dy	Horizontal spacing between elements in an array
I_0	Uniform amplitude excitation of an array
k	Angular wavenumber
β_x	Progressive phase shift between vertical elements in an array
β_y	Progressive phase shift between horizontal elements in an array

1 INTRODUCTION

The increasing complexity of systems getting engineered today poses a problem for traditional document-based systems engineering methods. Increasing complexity means more and more various documents are needed to capture all aspects of a system, and as the number of documents and their intricacy grows, their management becomes increasingly difficult and time consuming. Another problem with the document-based approach is that it doesn't have a single source of truth. This means that when, for example, a difference arises between two documents describing a system, it is not clear which one is right. This happens a lot since different documents are created and managed by people from different fields of expertise and levels of understanding.

This problem has arisen in the development of base stations and its various subsystems. One of these subsystems is remote radio head (RRH). This device is responsible for the radio frequency (RF) functionalities of the base station, for example filtering and amplification. The latest remote radio heads used in the fifth-generation mobile network technology (5G) base stations also include an integrated antenna module. These antennas are phased antenna arrays that enable a functionality called beamforming. By controlling the phases and amplitudes of the signals fed to the individual antennas of an array, the direction to and from which the whole antenna transmits and receives signals, can be controlled.

The systems engineering of phased antenna arrays at Nokia Solution and Networks Oy is document based, thus the first part of thesis demonstrates how the document-based engineering approach can be replaced with a model-based systems engineering (MBSE) approach. This involves the modeling of antenna array requirements, and its functional and logical architecture with a systems modeling language SysML in a modeling software application Cameo System Modeler.

The resulting artifact from these actions is a system model. This model is a system level abstraction of the system or component it represents, meaning it does not contain all the details of the antenna array design. This information is left to be captured in so-called domain specific models. Nevertheless, one of the main ideas of MBSE is that the system model would serve as the main source of truth for system level specifications and as a central hub into which the domain specific models are integrated. One such integrable domain is the analysis and simulation of phased antenna arrays.

Latest developments in electromagnetic simulation software and in the computing power of personal computers have made it possible to model the complex structures of phased antenna arrays and to simulate their electromagnetic characteristics and radiation performance. This enables the virtual design and analysis of an antenna array since the performance of its designs can now be verified before any actual physical manufacturing is needed.

Still, even with today's powerful computers, simulations for phased antenna arrays can take multiple hours. This introduces a need for models that take less time to simulate but can still be used to make approximations for the performance of a certain design of a phased antenna array. This is achieved with models at different fidelity levels. By starting the analysis from a low-fidelity model, a good understanding can be achieved of the designs alternatives that would meet the requirements of a certain application. This means that the amount of possible design can be minimized before moving on to analysis with the models that take much longer to simulate. Thus, this thesis demonstrates the creation of analysis models at three different fidelity levels that are used to analyse the effect of spacings between the individual radiating elements of an array.

A multi-disciplinary analysis and optimization (MDAO) tool ModelCenter enables the automation of various analysis and simulation tools. This software application is used in this thesis to automate the lowest fidelity model and to connect it to the system model, demonstrating the usage of a system model as the central hub for integration to domain specific models. Moreover, this software application includes powerful analysis and optimization tools, such as machine learning algorithms. These tools are used in thesis to find and to justify the optimum spacings between individual radiating elements in an antenna array operating in the frequency range from 3.4 to 4 gigahertz (GHz) based on the requirements specified and modeled in the system model. Finally, this resulting design data is uploaded to the system model, which demonstrates the multi directional connectivity between the analysis and the system model.

2 BACKGROUND

This chapter covers all the background information essential to understanding the context, and the theory behind the information presented in this thesis. This starts with considering the motivation and the principles behind model-based systems engineering, a modeling language that is used, and the enabling software.

Furthermore, the fundamental theory behind the function of phased antenna arrays will be covered, with an emphasis on the hardware aspect of beamforming and on performance evaluation methods and standards.

A substantial part of this thesis focuses on computer simulations of phased antenna arrays; therefore, fundamentals of electromagnetic simulation and software will be covered as well as the principle of different simulation fidelity levels. Additionally, another software used for post-processing simulation data will be introduced.

Another major aspect of this thesis is Multi-Disciplinary Analysis and Optimization (MDAO). The principles related to this methodology-in particular simulation automatization, design of experiments and optimization are explained. Finally, the software used for multi-disciplinary analysis and optimization and for combining the simulation and the model-based systems engineering domains to each other is introduced.

2.1 Model-Based Systems engineering

Understanding the concept of Model-Based Systems engineering (MBSE) starts with understanding what is meant by the word system. Let us consider some of the most common definitions of a system.

The International Council on Systems Engineering (INCOSE) has defined a system as “an arrangement of parts or elements that together exhibit behaviour or meaning that the individual constituents do not” [1].

The Institute of Electrical and Electronics Engineers (IEEE) standard 1471-2000 defines system as “a collection of components organized to accomplish a specific function or set of functions” [2].

For its simplicity and brevity, the methodology of Nokia uses the IEEE definition. But, all in all, a generalisation can be made that a system is a combination of elements that interact with each other and that are organized to achieve a purpose. This means that a system is not necessarily just a product, but that it can include people, environments, and some other supporting factors that might have a part in achieving the system’s purpose [3].

Systems engineering on the other hand has been defined by INCOSE to be “a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods” [4]. In this definition the term “engineered system” is used. This is a subtype of system and is defined by INCOSE to be “a system designed or adapted to interact with an anticipated operational environment to achieve one or more intended purposes while complying with applicable constraints” [5]. In this thesis the term “engineered system” is not used, since the system that is presented is an engineered system.

Traditionally systems engineering has been document-based. This results in a vast number of documents during the life cycle of the system, created by various stakeholders to capture the outcomes of their engineering actions. Since the stakeholders of the system are from various fields of engineering and expertise, diversity is required in the formats of documents that present the same information about the system, and it is obviously essential that all these

documents are valid and consistent with each other. Using a document-centric approach this gets very expensive and time-consuming, especially as the systems get more and more complex over time.

Model-based systems engineering (MBSE) is a solution to these problems. According to INCOSE it is defined as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [6]. This method's key difference to the traditional document-based approach is that the fundamental artifact resulting from the activities described in the definition is an integrated and harmonious system model.

System model is an abstraction of a system that is being engineered for a purpose and defined and evolved using model-based methods and tools. It is the resulting output from system specification and design actions performed by system engineers. The principal purpose of this fundamental artifact in MBSE is to enable the design and engineering of a system that fulfills its requirements and meets it comprehensive objectives. And as it enables this, the model also acts as a platform for communication between the system development team and other stakeholders involved in the system development. [7]

A system model includes specifications of the system and information on how both hardware and software components of the system are designed, analysed, and verified. This is because the system model can contain component interconnections and interfaces, component interactions and the related functions the components must perform, and various performance and physical characteristics or parametrics of the components.

All of these are achieved with different model elements that can represent requirements, design, test cases, design rationale or relationships between different parts of the system. These model elements and their cross-cutting relationships make it possible for the model to be viewed from multiple perspectives and aspects while maintaining uniformity between the views [7]. Different possible model elements, views of the system and their nomenclature depend on the modeling language that is used. The modeling language used in this thesis will be discussed in the next chapter.

The textual requirements for the system and often for the individual components are also captured in the model and traced to the higher-level system requirements. Requirements are usually imported and linked to the system model from an external requirements management system. This capability introduces the idea of a system model serving as a central hub for data integration with many other systems and models. This concept together with many different integrable data systems is illustrated in Figure 1.

Creating system and component level specifications that satisfy their requirements and perform their desired actions requires analysis. This is done by using analysis models that take advantage of computing power and computer software to predict or simulate the physical behaviour of a system or a component. The different analysis needs and requirements are captured in the system model and act as inputs for the analysis models. The outputs of the analysis models are the evaluated specification and its performance estimates. With this connection to the system model, the performance estimates can be compared to the requirements, and based on this the decision of the best possible specification option can be made. This connection and data exchange further extends the concept of the system model as the central hub for data integration.

The component-level specifications found in the system model act as inputs for in-depth design of the components in their respective engineering domain specific formats, for instance Computer-Aided Design (CAD) for hardware components and Unified Modeling language

(UML) for software components. In this way, system model acts also as a hub for data integration with specific engineering domain models.

All things considered, MBSE has numerous benefits, for instance improvement of quality by decreasing obscurities and expanding design integrity, and traceability between different engineering domains. It also improves productivity, enables automatization of parts of the process and improves knowledge transfer and reusability. [1]

A paper published in the 2017 Annual IEEE International Systems Conference [8], extends system and analysis modeling and introduces the concept of "Experimentable Digital Twins", which are virtual representations of components or parts of a system that, when used together, form a network that can be used in so-called Virtual Testbeds in order to develop, optimize and verify the functionality of the whole system.

MBSE has been already widely adopted in the aerospace and defense industries. A paper published in the 2014 IEEE Aerospace Conference [9], demonstrated how MBSE was applied to the simulation of a satellite mission that studies the formation of magnetic field-aligned electron density irregularities in the Earth's ionosphere. The system model created in the paper described the configuration and properties of systems and subsystems involved in the satellite mission. The system model was integrated to analysis models created in MATLAB that modeled functions of the mission's subsystems and states of the satellite itself.

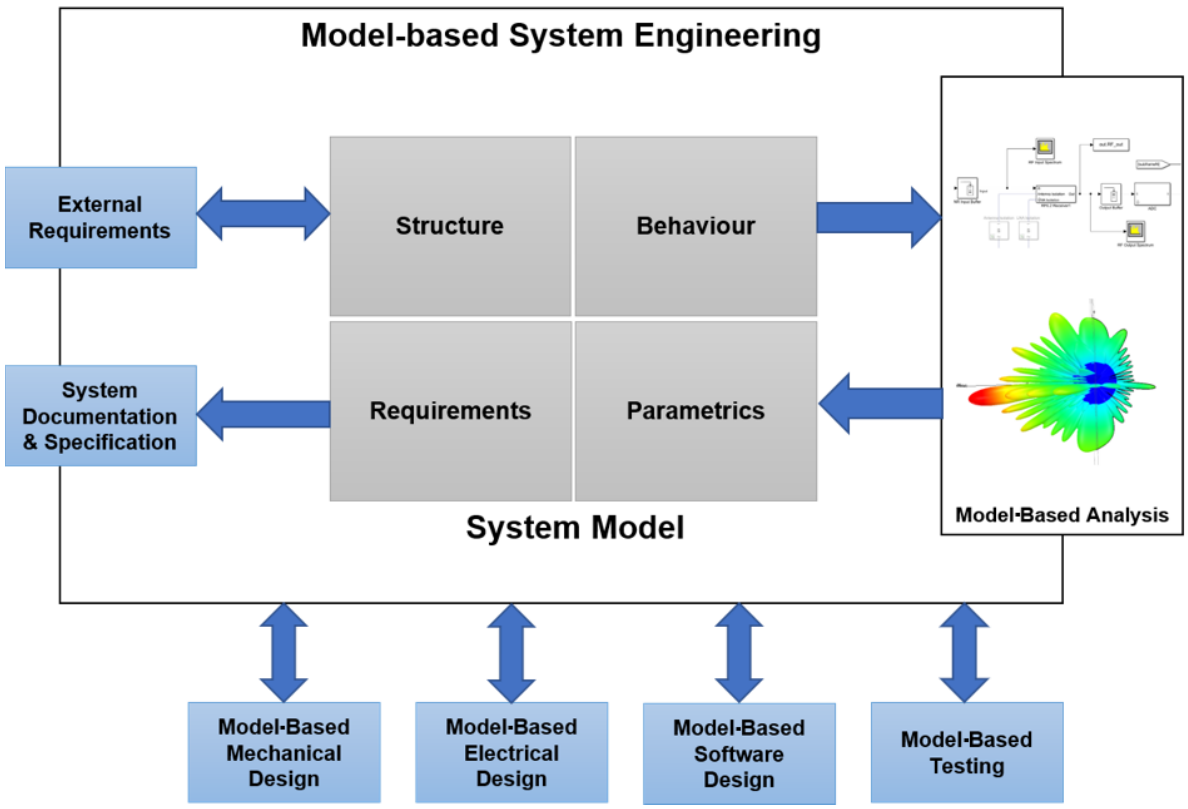

Figure 1. System model as the central hub for an integrated model framework.

2.1.1 Systems Modeling Language

Systems Modeling Language (SysML) is the most widely used modeling language used among model-based system engineers, and therefore the one used in this thesis. The Object Management Group (OMG) has defined SysML as "a general-purpose graphical modeling language for specifying, analysing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities" [10]. It is a graphical language that defines the semantic foundation for modeling system requirements, behaviour, structure and parametrics. These semantics consist of different possible model elements that are stored in a model repository and visualized on different possible diagrams with specific graphical symbols.

Elements called blocks are the basic units of structure in SysML. They can be used to represent hardware, software, or any other system element. Diagrams on the other hand are views of the model for a particular purpose. They enable modelers to enter model information, for instance blocks, into the repository, or to view certain information of the model from the repository. SysML includes nine different diagram types that are all meant to present model information in different aspects, enabling the model to be understood and analyzed from different perspectives. The name and purpose of each diagram is described in Table 1. The diagram's type determines what kind of model elements and their associated symbols can be visualized on the diagram. For instance, requirement diagrams can only display requirements and their relations to different functions or parts of the system, but not relations between parts. On top of diagrams, SysML comes with support for tabular, matrix and tree views of the model. [7]

Table 1. SysML diagram types and their purposes

Diagram Name	**Purpose**
Requirement diagram	Presentation of requirements and their hierarchies and relations to other elements in the model.
Activity diagram	Description of data flow and control between a system's activities.
Sequence diagram	Description of interactions between collaborative parts of the system.
State machine diagram	Description of state transitions and actions of the system or its parts in response to events.
Use case diagram	High-level description of functionality in terms of how the system interacts with external entities.
Block definition diagram	Description of system hierarchy and structural element classification.
Internal block diagram	Description of interconnections and interfaces between a system's structural elements.
Package diagram	Presentation of the model organization
Parametric diagram	Presentation of constraints on system property values and serves to integrate with external engineering analysis models.

2.1.2 Model-based Systems Engineering Methodology

A modeling language like SysML alone does not specify how the system model is created and modeled. A modeling method is essential for providing the system engineers a road map and guidelines that ensure that the modeling produces expected results.

The OMG uses the following definition of MBSE methodology: “The collection of related processes, methods, and tools used to support the discipline of systems engineering in a “model-based” or “model-driven” context” [1]. With a methodology MBSE can be tailored to fit a specific scope of development and for systems of specific fields of industry.

The system model introduced in this thesis is built with a methodology grounded on the learnings and best practices in public and the deep understanding of the challenges Nokia has faced and is facing in the development of more and more complicated systems.

2.1.3 Cameo Systems Modeler

Cameo Systems modeler, also called MagicDraw, is the system modeling tool used in this thesis. It is developed by No Magic Inc. and it is one of the industry leaders in providing an environment for MBSE. It allows defining and visualizing systems with SysML standard compliant model elements and diagrams.

A plug-in called “DataHub” allows integration with different databases. In this thesis it is used to integrate a web-based requirements management tool DOORS Next by IBM to the system model. In this way the requirements defined and managed in DOORS can be brought to and used in the system model. Another plug-in called “Simulation Toolkit” allows integration between the system model and analysis models.

Another closely related software used in this thesis is Teamwork Cloud (TWC), which is from the same company and is used together with Cameo Systems Modeler. It is a repository for collaborative development of system models and model version storage.

With TWC, multiple engineers can read or even modify the same model that is stored on a cloud server. Changes to the model are done by locking certain parts of the model for editing, and after the changes are done, the user commits the changes to the server and the model updates for everyone. The locking feature allows the whole model to be edited by multiple people at the same time, but only if they are editing different parts of the model. The server keeps track of the committed changes and they can be reverted if necessary.

2.2 Phased Antenna Arrays

Phased antenna arrays are antennas composed of several radiating antenna elements. Each of them is fed with a coherent radio signal, but the phase or time-delay of the signal fed to each individual element is controlled. This enables the repositioning or scanning of the direction of the main beam, which is the direction where the radiated power density of the antenna is the highest. This is an essential feature in modern wireless communication systems since it enables radio transceivers to adapt to the positioning of other uplink or downlink transceivers. For instance, a base station equipped with a phased antenna array would be able to direct its main beam or the biggest part of its power to a direction where most of its users are. This functionality is called beamforming and its implementation involves various aspects in both hardware and software.

Besides beamforming, the radiation characteristics of an antenna array can be controlled with the geometrical configuration of the array and the relative displacement between the individual radiating elements [11].

Antenna arrays used in modern telecommunications are planar arrays. In planar arrays, the individual radiators are positioned along a rectangular grid, with a certain amount of radiators placed horizontally and vertically within certain distances from each other [11]. This results in four different parameters that can be used to describe an antenna array. These are as follows: (1) the number and (2) the spacing of vertical radiating elements; and (3) the number and (4) the spacing of horizontal radiating elements. Planar array and its parameters are shown in a spherical coordinate system in Figure 2, where M is the number of vertical elements and dx the vertical spacing, N is the number of horizontal elements and dy the horizontal spacing.

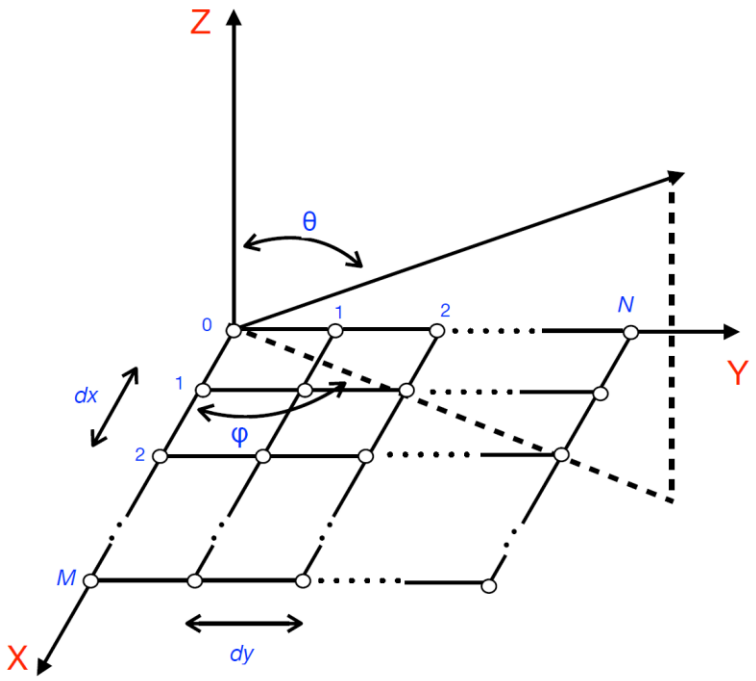

Figure 2. Geometrical parameters of planar antenna arrays.

If all the elements in an array are identical, the total electric field of the antenna array is equal to the field of its single element positioned in the origin, multiplied by a factor known as the Array Factor (AF).

$$E(total) = [E(single\ element\ at\ origin)] \times [AF] \quad [1]$$

The array factor is a function of the number of elements in each direction in an array, their respective spacings, and their relative magnitudes and phases. The array factor is quite complex as it encompasses everything, but it will be simplified significantly if the array's elements have identical amplitudes and spacings. In this case, the array factor for the entire planar array is written as

$$AF = I_0 \sum_{m=1}^{M} e^{j(m-1)(kd_x sin\theta cos\phi + \beta_x)} \sum_{n=1}^{N} e^{j(n-1)(kd_y sin\theta sin\phi + \beta_y)} \quad [2]$$

where I_0 is the uniform amplitude excitation of the array, k the angular wavenumber and β_x and β_y the progressive phase shifts between the elements along the two axes. [11]

First approximation of the antenna array's performance can be made from just calculating its array factor, but this is not very accurate, since knowledge of the single element's radiated electric field is also needed. Furthermore, coupling between the individual elements must be also considered for accurate analysis.

2.2.1 Microstrip Radiators

One of the most common types of radiators used in modern telecommunication arrays are microstrip antennas. These elements consist of a conducting patch on a grounded substrate. The patch can be of any shape, but rectangular patches are the most widely used. This is because they have better performance and their analysis and fabrication are more straightforward. [11]

An example of a simple rectangular microstrip antenna and its most common design parameters are illustrated in the Figure 3. The parameters are the length, width and thickness of the patch element, and the height and relative permittivity of the substrate. The optimal values for these depend on the desired operating frequency band and the desired radiation characteristics of the antenna.

With the knowledge of both the individual radiator elements and the geometrical configuration of the array, an in-depth analysis can be made.

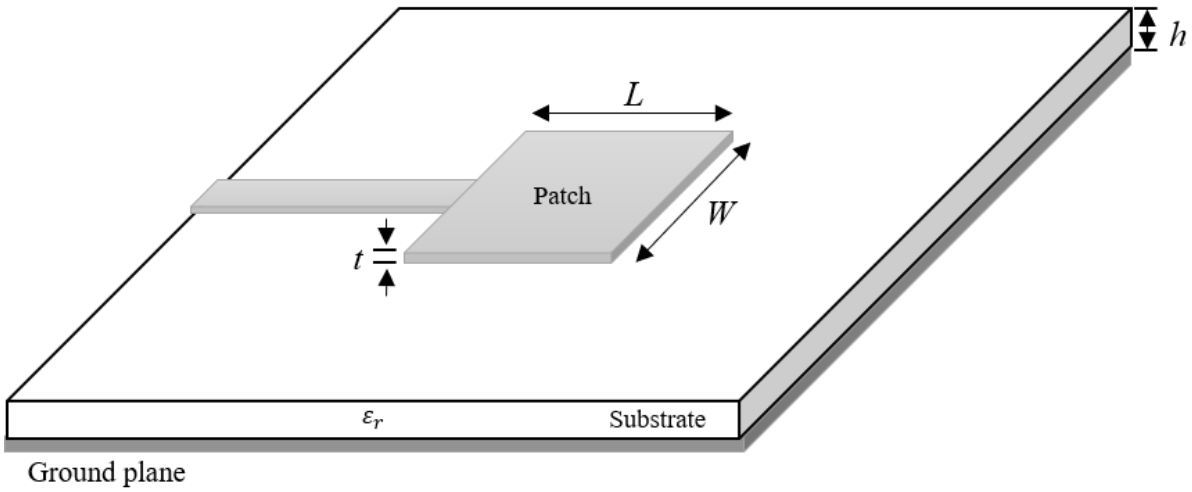

Figure 3. Rectangular microstrip antenna.

2.2.2 Performance Validation

Various parameters are necessary to describe and validate the performance of an antenna array. Most of these parameters are derived from a so-called radiation pattern that represents the antenna's gain in three-dimensional space. Gain is defined as a ratio of its radiation intensity to the radiation intensity that would be obtained if the power accepted by the antenna were radiated isotropically. As this is a ratio with a specific reference, its unit is defined as decibels per isotropic antenna (dBi). [12]

Antenna arrays are designed to operate at a certain frequency band, and over a certain steering range for the main beam. Therefore, the performance must be evaluated so that both two aspects are considered. This is done by simulating or measuring the radiation pattern at different frequency points inside the operating frequency band, and when the beam steered to

different angles. Choosing the amount of different frequency points and steering angles is a trade-off between simulation or measurement time and accuracy.

The next generation mobile networks alliance (NGMN) has provided recommendations on Standards for base station antenna performance validation (BASTA). These standards specify different statistical analysis methods for the performance parameters and how each of them is defined. The definitions of different parameters used in this thesis are discussed in the following paragraphs.

One of the most important parameters is the main beam's peak gain or the maximum gain at electrical boresight, i.e. when no steering is applied.

One way to evaluate the antenna array's steering performance is its peak gain drop, which is a measurement of the drop in the peak gain when the main beam is steered to a certain angle, compared to the gain achieved at electrical boresight.

Another important parameter is half-power beamwidth (HPBW) that is the angular width of the main beam. This is calculated by evaluating the points of the main beam in which the gain has dropped three decibels (dB) from the maximum. The angle between these points is the HPBW, and it is evaluated for both horizontal (azimuth) and vertical (elevation) radiation patterns. [13]

A parameter called Front-to-back ratio (FBR) is used for evaluating how much energy is radiated backwards from the antenna. It is specified in dB and defined by NGMN BASTA as "the ratio of power gain between the beam peak and the highest total power value in the 60° angular region of the azimuth cut contained between two boundaries, each 30° distant from the axis corresponding to the nominal direction ± 180°." [13]

Grating lobes are strong sidelobes that can appear in the radiation patterns of antenna arrays. Their levels are measured with a parameter called Grating lobe suppression (GLS). This is defined as the difference between the gain of the main lobe and the gain of the grating lobe, and it is specified in dB.

The radiation pattern also contains sidelobes, i.e. lobes other than the main lobe. Their levels are evaluated with a parameter called Sidelobe suppression (SLS). This is defined as "gain difference between the main beam peak and the highest gain level amidst all the sidelobes". This parameter is specified in dB and usually evaluated separately for the sidelobes that are above or below the main lobe. These are respectively called upper and lower sidelobe suppressions. [13]

If the antenna supports two orthogonal polarizations, the correlation between them is measured with Cross-polar discrimination (XPD). This is defined as the "ratio of the co-polar component of the specified polarization compared to the orthogonal cross-polar component". This parameter is specified in dB and usually evaluated, both at the main beam's peak and within its HPBW. [13]

In addition to parameters derived from the radiation pattern, there are parameters derived from the electrical characteristics of the antenna array. One of them is Return loss (RL) that is "the ratio (in linear unit) between forward and reflected power measured at the antenna port over the stated operating band" [13]. This parameter is specified in dB and it is an indicator of how good the power delivery is from the antenna's inputs to its radiated output. A high value means that there is a good impedance match between the transmission lines and the array's radiators.

Another parameter is isolation between the array's radiators. It is the ratio between the power delivered to one of the radiators and the power detected from the port of another radiator. This parameter is specified in dB and it describes how much of the power delivered to one radiator leaks to another radiator's ports.

2.2.3 Antenna Array Analysis

As previously mentioned, in-depth analysis of antenna arrays requires knowledge of the structure of individual radiators and the geometrical configuration of the array. Modeling and calculation of the electromagnetic properties of such complex structures requires a lot of computing power.

Electromagnetic (EM) simulation computer software has been developed to tackle this problem. The most advanced software has a graphical user interface (GUI) for creating the three-dimensional (3D) structure of the radiating elements and constructing an antenna array from them. After the 3D structure and the materials it consists of have been defined, a simulation of the array's electromagnetic end electrical properties can be initiated.

One such software used in this thesis is Ansys HFSS. This EM simulator solves the electrical and magnetic fields of the antenna by trying to find the numerical solutions of Maxwell's equations in differential form [14].

There are many different mathematical methods for this, but the most widely used is Finite Element Method (FEM), which is derived from Maxwell's curl equations and works in the frequency domain. This method works by dividing space into tetrahedron elements that represent the smallest volume entities of the model. The electric and magnetic fields inside the tetrahedrons are expressed as a set of polynomial functions that yield a large linear system of equations when their variation is set to zero. A matrix eigenvalue is then solved to find the electric and magnetic fields at all nodes at each desired frequency. [14]

This type of solver is especially effective in solving problems with periodicity such as phased antenna arrays [14]. The results of a successful simulation are the electric and magnetic fields of the antenna array, from which the EM simulation software is able to produce more practical results such as 3D radiation patterns and s-parameters of the antenna array at each of the simulated frequencies.

Obtaining values for the performance parameters defined in the last chapter requires further data post processing. One tool for this is MATLAB, which is a widely used programming and numeric computing platform. With it, a custom script can be created for manipulating the results data coming from the EM simulator and performing statistical analysis on it.

2.3 Multi-Disciplinary Analysis and Optimization

Multi-Disciplinary Analysis and Optimization (MDAO) is an engineering method that aims to combine analysis models from different disciplines or engineering domains and to use them to analyse, and ultimately to find out and to justify the best possible design configuration for the system. The design variables of a system act as inputs, and different metrics that, for instance, tell how well the system performs, act as outputs. These factors are initially defined in different domain specific analysis tools, but in MDAO these tools can be combined and integrated, and their analyses executed at the same time.

MDAO is especially powerful if it can be integrated with MBSE. With this integration different parameters specified in the system model, can be connected to their analysis model counterparts, the inputs, and outputs in MDAO. This connection enables uploading new design configurations, achieved with MDAO, back into the system model. In addition to this, MBSE integration enables connection to the system or component requirements specified in the system model. This enables better evaluation of different design configurations, since the requirements can be considered at every step of analysis and optimization.

All aspects considered, the main goal of MDAO is to help engineers designing a multidisciplinary system to come up with the best possible design, and to form a connection between domain specific analysis models and the system model.

2.3.1 Design of Experiments

Analysis of the system is usually started by performing multiple rounds of Design of Experiments (DOE). This method involves performing and planning experiments on the design, so that their results can be analysed to draw statistically valid conclusions. In practice this means making repeated executions of a simulation with varying sets of inputs to see the effect different design variables have on different metrics of the system. This enables engineers to understand which of the design variables have the most impact on the simulation outputs and what kind of correlation exists between each of them.

When the inputs of interest, their boundaries, and the amount of experiments are all specified, different mathematical techniques can be used to automatically create a set of experiments for the inputs. Most techniques aim to minimize the amount of experiments, while withholding the ability to make statistically valid conclusions on the effects of the inputs.

After a DOE is finished, its results can be visualized, making it easy to analyse the behaviour of different metrics with design parameters. This is illustrated with an example in the Figure 4, in which each point represents a unique design alternative.

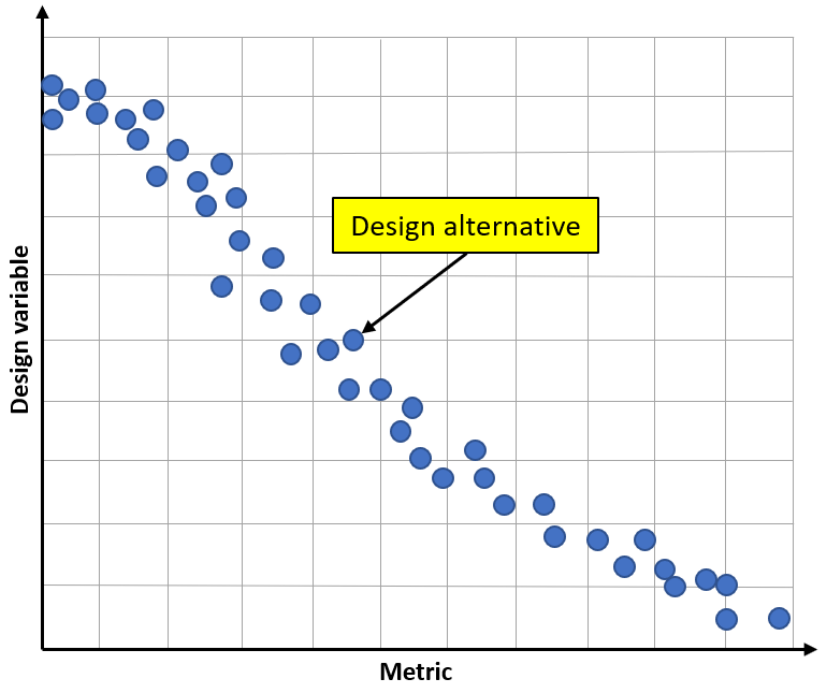

Figure 4. A graph of DOE results.

2.3.2 Optimization

Optimization is a major part of MDAO. It involves using deep-learning algorithms to find the optimum design. It starts with defining the inputs (design variables) that are to be optimized. Then each input's boundary, meaning its lowest and highest possible value, are defined. After these, objectives for the optimization are defined. This consists of choosing different simulation

outputs or metrics and defining their goals, either aiming to maximize or minimize them, or to reach a certain value. On top of this, different constraints can be defined for the optimization. These are usually different requirements that, for instance, specify whether some metric or simulation output can be allowed to exceed a certain threshold. Designs that do not meet these constraints are automatically avoided by the optimizer and not considered.

Finally, an algorithm for the optimization is chosen. There are various algorithms, but only one is utilized in this thesis, a genetic search algorithm called Darwin. This algorithm is used because it can solve optimization problems with multiple objectives and constraints. Furthermore, it allows defining tolerances for design variables, i.e. the number of decimal places the optimizer will consider for a certain variable. As the name suggests, the function of this algorithm is inspired by biological evolution observed in nature. More in-depth information about multiobjective genetic algorithms and challenges in their usage in solving complex real-world problems can be found from a paper published by Yaochu Jin and Bernhard Sendhoff in the IEEE Computational Intelligence Magazine Vol. 4, Issue 3 [15].

The algorithm randomly generates a certain number of designs. The number is determined by a configurable parameter called population size. Then each design in the population is given a fitness score which is calculated according to the optimization objectives. The design that meets the objectives the best is ranked with the highest score, etc. Different design parameter values that make up the designs are considered as genes. The designs with the highest scores are chosen to be used to create a new generation of designs by mixing their genes with a small amount of random variation.

This formation of new generations continues as long as the new generations are different from each other. If a certain amount of generations has been generated without significant differences, the optimization algorithm converges i.e. is finished. If the optimization has multiple objectives, a single definite optimum design alternative cannot be reached, and therefore the optimization algorithm is searching for a series of best designs. The designs in this set are called pareto optimal designs, which are the final outcome of the optimization.

Two parameters are used to configure the convergence. These are the amount of generations without improvement and the minimum percentage change of pareto population required for improvement. The latter specifies how much the pareto-optimal designs must change across successive generations. If less than a specified percentage of them do not change, the generations without improvement counter is incremented.

In similar fashion to DOE, optimization results are also visualized with various graphs. In these graphs, different objectives of the optimization are compared to each other. This results in a visualization of the pareto front. In order to further illustrate this concept, an example of a graph with pareto front visualized is given in Figure 5.

The pareto front is made up of best possible designs that cannot be improved in any of the objectives without degrading another one. As a result, trade-offs are necessary. However, in most cases some objectives are more important than others. This additional subjective preference information is used to make the final decision of the best possible design.

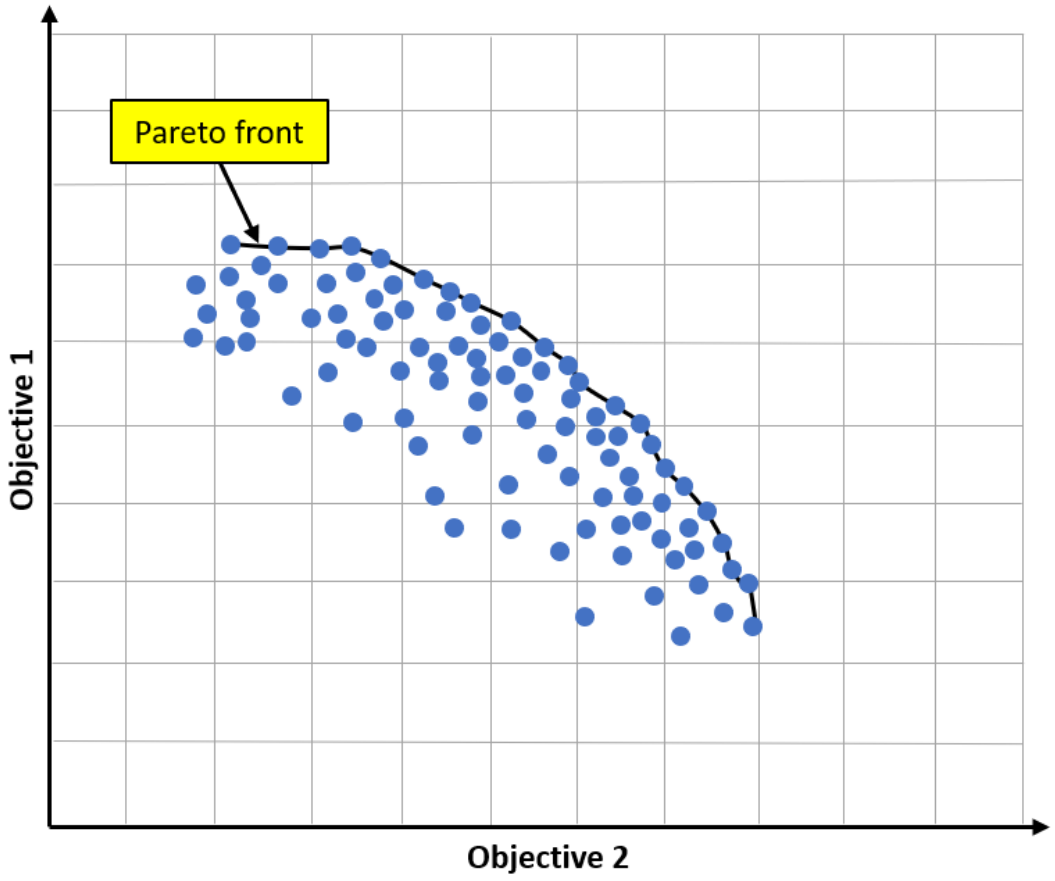

Figure 5. Pareto front visualized in a comparison of two competing objectives.

2.3.3 ModelCenter

ModelCenter from Ansys, inc. is the MDAO software application used in this thesis. This tool enables the automatization of simulations tools, models, and scripts. These automized simulation components can then be combined to create complex analysis workflows. Furthermore, DOEs and machine learning driven optimization can be performed on the created analysis workflows. On top of these features, ModelCenter is the only widely commercially available MDAO application that has a built-in capability for connecting with system models created in Cameo Systems Modeler, and for this reason it is used, and its features explored in this thesis.

The main feature of ModelCenter is the ability to make complete analysis workflows from the automated tools and their respective models with a drag-and-drop user interface. The analysis workflow can be described as a flowchart that specifies the order, as well as the conditions under which the automated tools are to be executed. And much like flowcharts, workflows may contain parallel branching, if-then statements or loops.

A major part of the workflow creation is the usage of a link editor. It is used to specify the data that should be transferred from one simulation component to another. This is required when, for instance, some component requires information from the output of another component, i.e. analysis result. In this case the output data of the first component is connected to the input of the next component and the models are executed consequently in the workflow.

Simulation tools are automated with ModelCenter with either custom scripts or in-built connectors. At the moment, only the most common tools, for example MATLAB and Excel have in-built connectors, so their automation can be accomplished with a simple point and click interface. These in-built connectors are able to automatically recognize the input and output parameters from the components, or they can be manually chosen in the user interface.

More advanced tools like Ansys HFSS, require scripting. Python, which is one of the most common programming languages today, is supported by ModelCenter. Any tool can be automated with ModelCenter by creating Python scripts that modify and execute analysis models in a simulation tool. However, these Python scripts do not work jus as they are, but they require a ModelCenter specific header and file type. In the header, the chosen programming language, and its input and output variables are all specified. Finally, the file type of the script is changed to ".scriptWrapper". Now it can be automatized by simply dragging and dropping it to an analysis workflow, where it is then seen as a ScriptWrapper component.

In addition to this, any Microsoft Windows executable program or a batch script that uses textual input and output files can be easily automated with the QuickWrap feature of ModelCenter. The executable file or the batch script and their respective input and output files are all selected in the user interface of QuickWrapper. The input and output parameters are automatically extracted from their respective files. This creates an automated simulation component. The inputs of the component can now be configured through ModelCenter and its outputs are returned to ModelCenter.

In Figure 6, an example of an analysis workflow is given. It consists of two QuickWrapper components that are executed consequentially, and of a MATLAB component and a ScriptWrapper component that are executed in parallel after the QuickWrap components.

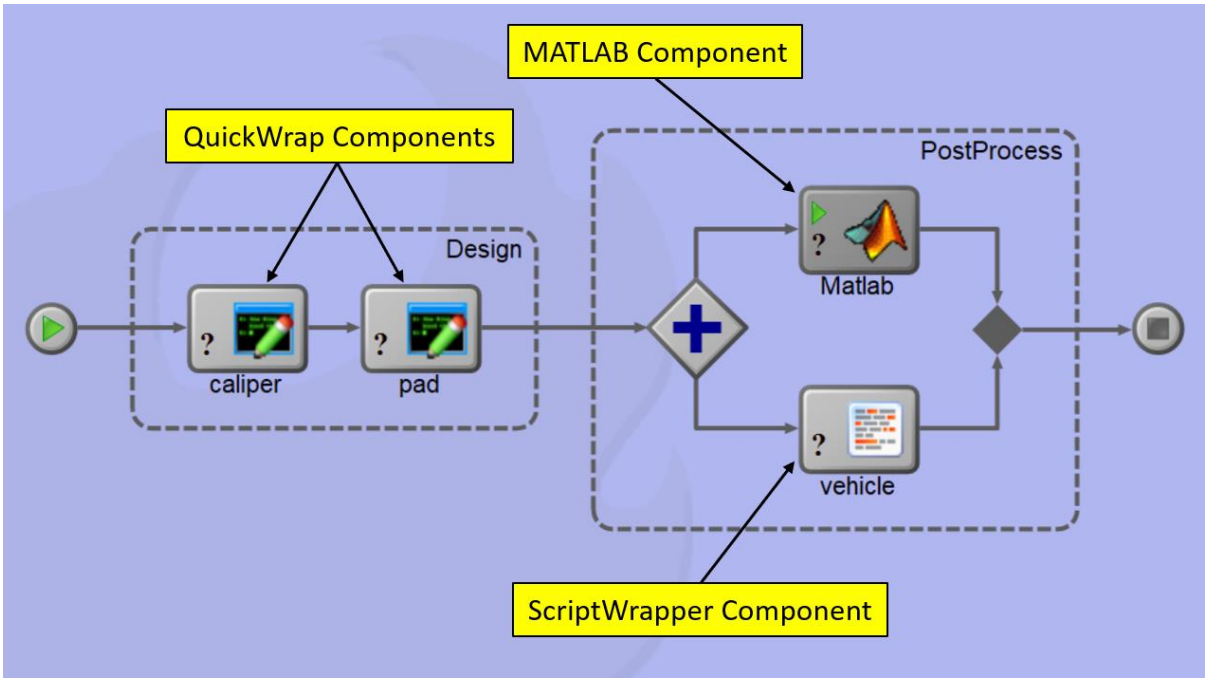

Figure 6. An example of an analysis workflow created in ModelCenter.

Like it was mentioned in the beginning of this chapter, analysis workflows can be connected to system models. With this feature ModelCenter bridges the gap between system modeling and analysis domains, and thus becomes a part of the previously mentioned integrated model framework. But even before workflows can be connected to a system model the analysis

engineers need information on the analysis needs. The parameters, both design variables and different metrics defined in the system model, are information that analysis engineers use to build their analysis workflows. Design parameters defined in the system model have to exist as inputs, and the metric or performance parameters have to exist as outputs in the workflow.

After the workflows and its components have been built to match the parameters in the system model, a connection can be established. This means that any values for the parameters defined in the system model can be automatically used in the workflow. Furthermore, if parameters are connected to requirements in the system model, they can be automatically brought to the workflows in ModelCenter and used to perform requirements conformance analysis. In practice this means that when DOEs are performed, the unsatisfied requirements can be automatically highlighted for different design configurations. And in optimization, the requirements can be used as constraints that guide the algorithm to find designs that are not only pareto optimum, but also satisfy all the requirements.

DOEs and optimization are configured in ModelCenter's graphical user interface. The DOE tool, seen in Figure 7, contains slots for design variables and response variables into which they can be simply dragged and dropped. The design table, which specifies the different experiments that make up a DOE, can be created manually or automatically with different algorithms that can be chosen from a drop-down menu under "Design". When the DOE is initiated, the workflow is executed repeatedly according to the design table. The values of the design variables and the response variables at each iteration are stored in a table.

Another widely used MDAO software is modeFRONTIER from ESTECO. In a paper published in the 2008 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems [16], it was used to automate a seven element antenna array analysis model created in a 3D EM simulation tool CST Studio Suite, and to perform Multi-objective optimization on the antenna array design.

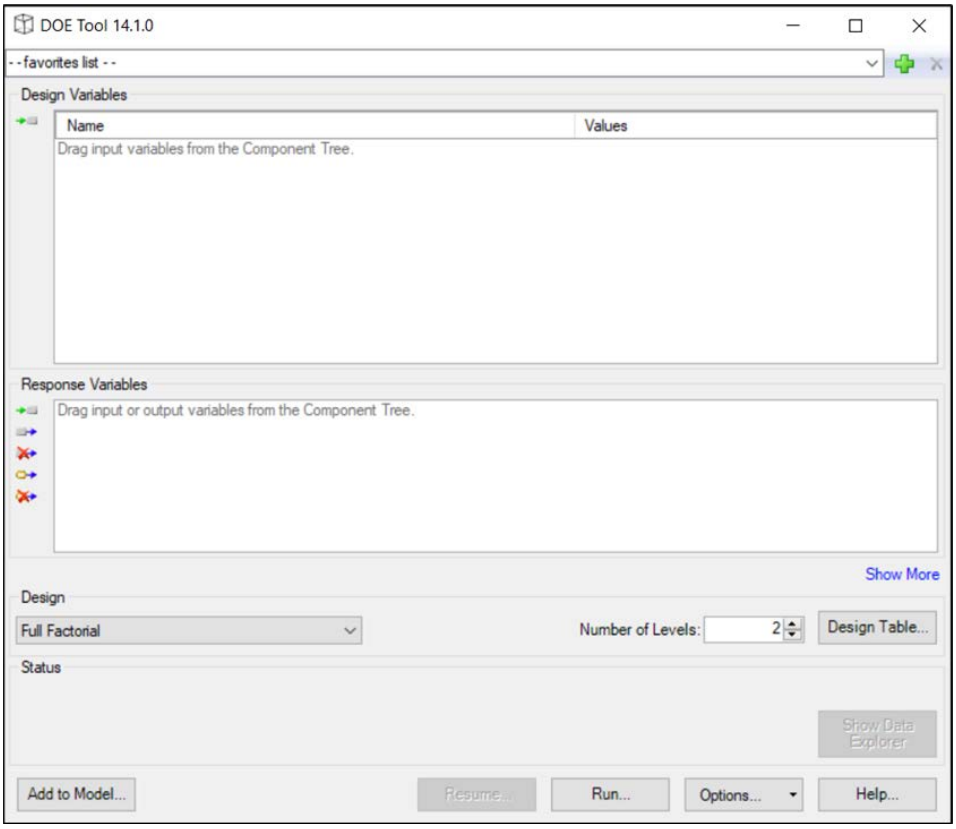

Figure 7. DOE tool GUI.

Optimization tool on the other hand contains slots for the objectives, constraints, and design variables, as seen in the Figure 8. Like in the DOE tool, the variables are defined by dragging and dropping them to their slots. The optimization algorithm is finally chosen from a drop-down menu in the lower part of the GUI under “algorithm”. When the optimization is started the workflow is executed repeatedly by the algorithm. The values of the design variables, constraints, and objectives at each iteration are stored in a table.

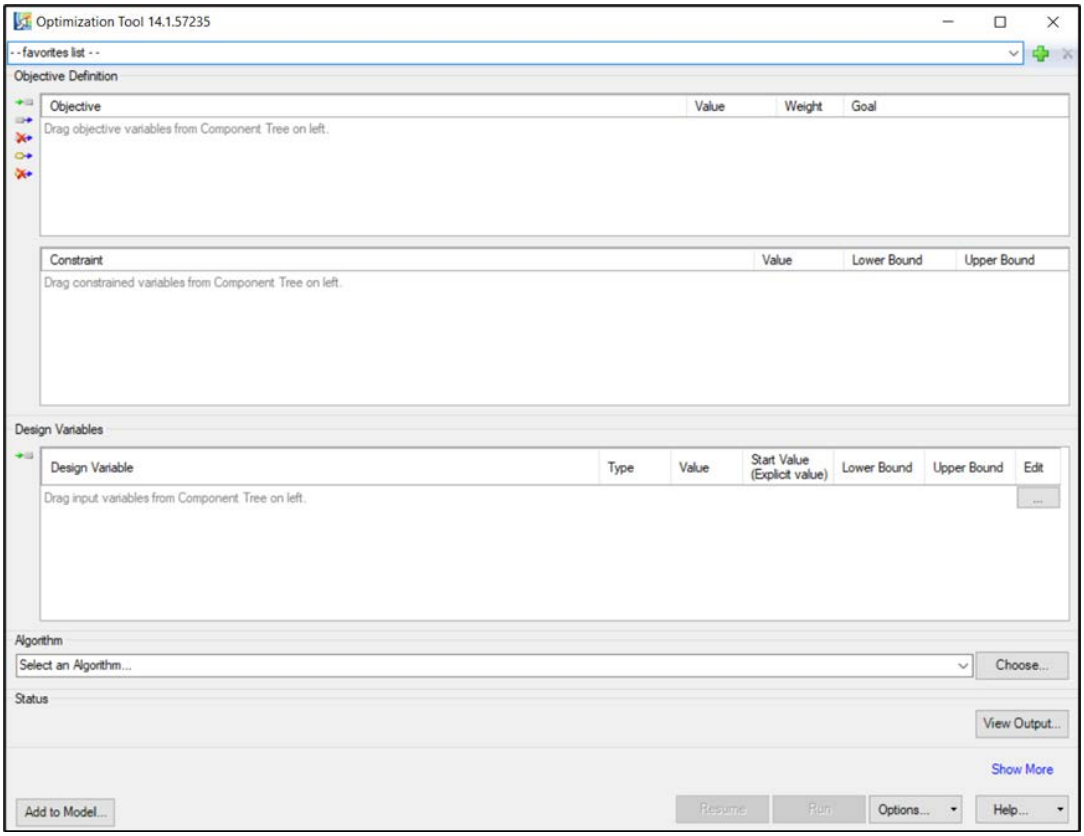

Figure 8. Optimization tool GUI.

Designs achieved with DOEs or optimizations can be uploaded back into the system model. ModelCenter does this by creating a SysML element called instance. This element contains the values of the parameters that were connected to the system model earlier and make up a chosen design. This design information can then be used in Cameo Systems Modeler for generating design specifications and reports, or for further higher-level analysis.

3 PREMISES OF THE ANTENNA ARRAY USE CASE

This chapter will introduce the starting point of the antenna array use case for Model-Based Systems Engineering and Multi-Disciplinary Optimization. The goal of the use case is to find out how telecommunications antenna arrays could be modeled with SysML, and how the analysis and optimization of their design could be done while being connected to the system model. An example of this is provided with a planar antenna array that is to be used in a 64-transceiver massive MIMO (Multiple-input and multiple-output) remote radio head (RRH) operating in the frequency range from 3.4 to 4 GHz. The design problem is to find the optimum vertical (elevation) and horizontal (azimuth) element spacings for the array. The use case is based on the ideas and practical suggestions presented in a webinar held by Phoenix Integration and Northrop Grumman [17].

The example can be divided into four main parts. First is the creation of a system model for the antenna array, which consists of modeling its requirements, and functional and logical architectures with SysML in Cameo Systems Modeler. Second is the creation of analysis models in Ansys HFSS and MATLAB. The third one is the building of an analysis workflow and connecting it to the system model. Finally, a DOE and optimization are executed. These result in trade studies and eventually in finding the best possible design option for the antenna array, which is finally uploaded to the system model.

3.1 Radiating Elements

The example starts from a setting in which the structure and the number of vertical and horizontal radiating elements of the array have already been decided. The elements used to build the array are rectangular air-filled dual coaxial probe fed patch antennas. This means that there is a layer of air on top of the substrate of the patch, and that the signal is fed to it through two coaxial connectors. These two connectors enable the patch to transmit and receive radio waves in two orthogonal linear polarizations.

The printed-circuit board (PCB) on which the radiator lies on, has a differential feed line network that divides the signal coming from each coaxial connector into two legs that extend through the air and connect to the patch. One pair of legs is responsible for producing a linearly polarized signal slanted 45° and the other pair for a linearly polarized signal slanted 45° to the opposite direction. The structure of the element is visualized in the Figure 9.

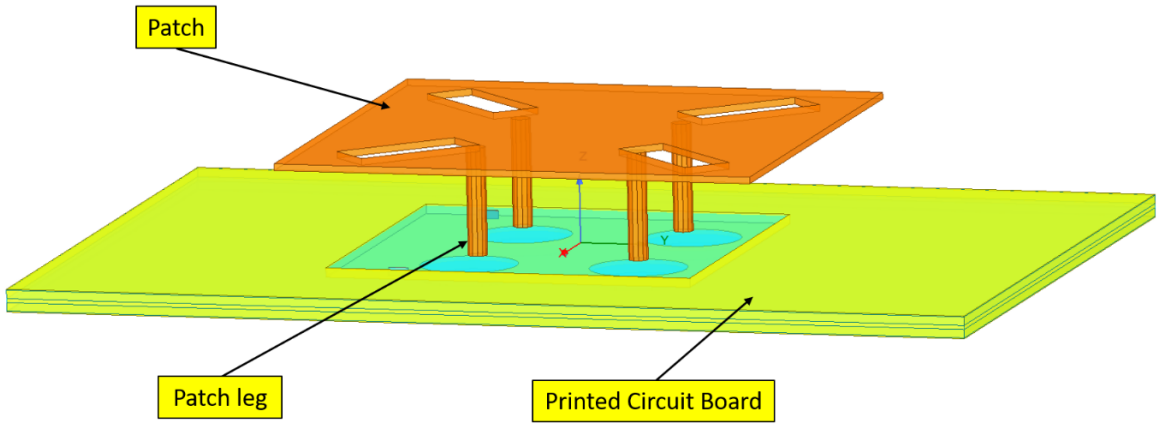

Figure 9. The structure of an individual radiating element.

Like previously mentioned, the array constructed from these elements is meant to be used in an RRH with 64 parallel transceivers. This means that the signal comes to and from 64 ports of the antenna, which in dual polarized antenna are further divided into 32 ports for one polarization, and 32 for the another. Various studies conducted at Nokia have concluded that the best way to build an antenna array for a 64-trasceiver RRH is to construct it from sub-arrays consisting of three vertical radiating elements, which are sets of radiating elements that are all fed with the same signal coming from the same port.

Since the radiating elements all have two ports, one for each polarization, 32 sub-arrays are required, which are organized by placing four sub-arrays vertically and eight horizontally. This results in an antenna array that has 96 radiating elements in total, twelve elements vertically and eight horizontally. The sub-array structure is implemented with a feedline network that divides the signal from the coaxial main ports to the ports of the individual radiating elements. The Figure 10 further illustrates how a sub-array is constructed from the dual-polarized radiating elements and how two transceivers are connected to each sub-array. Different colours are used in the figure to highlight the two orthogonal polarizations in the radiating elements. In order to separate the polarizations of a sub-array from each other, another term called array element is used in this thesis. In the case of dual-polarized radiating elements, each sub-array consists of two array elements, one for each polarization.

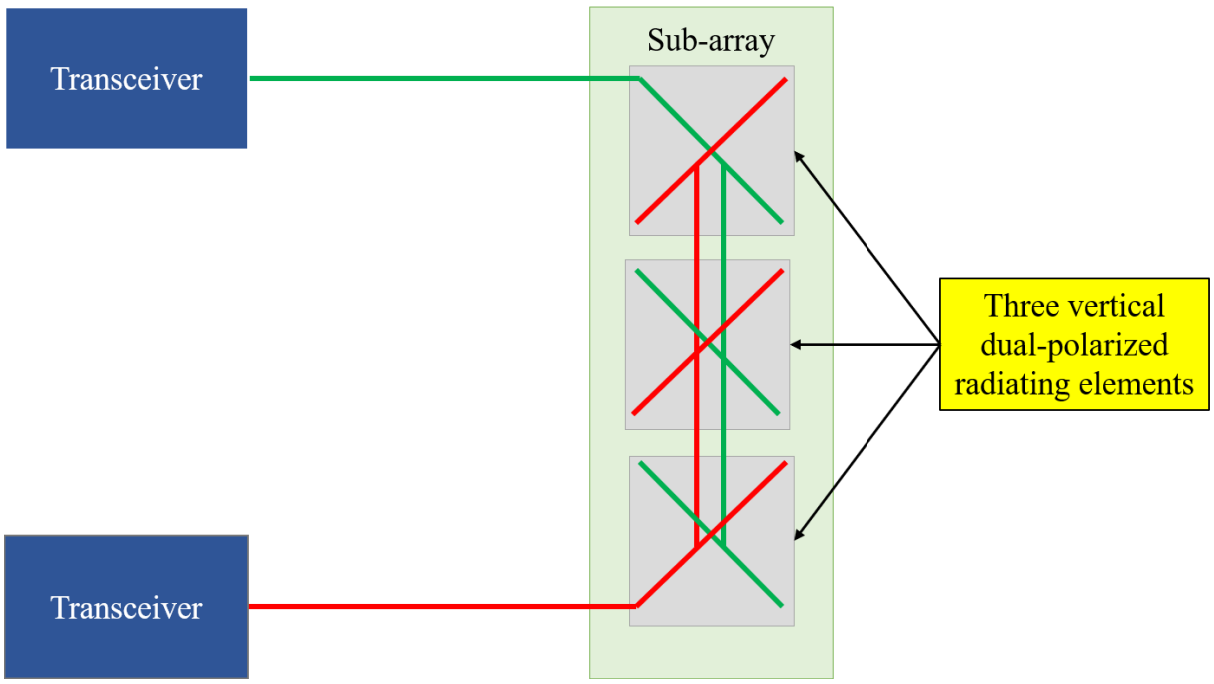

Figure 10. A sub-array connected to two transceivers, one for each polarization.

3.2 Requirements

The antenna array has to be designed so that it fulfills its requirements. Generic antenna array requirements were created for this thesis in the previously mentioned requirements management tool. These requirements are measurable values for different antenna performance parameters that were covered in the chapter 2.2.2. The values are calculated with different BASTA analysis methods over measurements done at multiple frequencies and steering angles. The descriptions of the requirements and their BASTA analysis methods are listed in the Table 2.

Table 2. Antenna array requirements and their analysis methods

Requirement Description	**BASTA Analysis Method**
Antenna array peak gain at electrical boresight shall be at least 24.5 dBi.	Mean, calculated with linear values.
Antenna array peak gain drop over 90° steering range shall be no more than 3 dB.	84th percentile value.
Antenna array peak gain drop over 120° azimuthal envelope beamwidth shall be no more than 7 dB.	84th percentile value.
Antenna array azimuthal Half Power Beamwidth (HPBW) at electrical boresight shall be at least 11° and no more than 14°	6.7th and 93.3th percentile values.
Antenna array front-to-back ratio (FBR) over azimuth and elevation steering range shall be at least 25 dB.	16th percentile value.
Antenna array Grating Lobe Suppression (GLS) inside 180° azimuth opening angle, over 120° azimuth steering range shall be at least 6 dB.	Worst case value.
Antenna array Cross Polar Discrimination (XPD) at beam peak, at electrical boresight shall be at least 20 dB.	16th percentile value.
Antenna array Cross Polar Discrimination (XPD) within Half Power Beamwidth (HPBW) of the main beam and over azimuth and elevation steering range shall be at least 10 dB.	16th percentile value.
Antenna array elevation Half Power Beamwidth (HPBW) at electrical boresight shall be at least 5.5° and no more than 7.5°.	6.7th and 93.3th percentile values.
Antenna array *upper* elevation Side Lobe Suppression (SLS) over full elevation steering range shall be at least 10 dB.	16th percentile value.
Antenna array *lower* elevation Side Lobe Suppression (SLS) over full elevation steering range shall be at least 10 dB.	16th percentile value.
Antenna array *upper* elevation Side Lobe Suppression (SLS) over full elevation steering range +/-2° shall be at least 6 dB.	16th percentile value.

Antenna array *lower* elevation Side Lobe Suppression (SLS) over full elevation steering range +/-2° shall be at least 6 dB.	16th percentile value.
Isolation between array element ports shall be at least 20 dB.	Worst case value
Array element Return Loss shall be at least 14 dB.	Worst case value

4 ANTENNA ARRAY SYSTEM MODELING

The use case starts with the system modeling work of the antenna array with SysML in the Cameo Systems Modeler tool. The work can be divided into three parts, which are modeling of the requirements, modeling of the functional architecture, and modeling of the logical architecture.

The system itself in this use case, is the whole remote radio head, but only its array antenna component is modeled in detail. In this way the model could be extended in the future to include other components of the RRH as well.

The main purpose of the system model created in this use case is to demonstrate how a component of a system, the antenna array, could be modeled with SysML, so that it can be used as the central hub of the integrated model framework for analyses done with the MDAO tool ModelCenter. Since only one component of the system is modeled, many aspects of system modeling have been left out.

4.1 Model Structure

Model repositories of the system models need a well-defined structure that makes them easy to read and logical. Therefore, the structure defined by the MBSE methodology of Nokia and the one used in this use case will be now introduced.

What is considered as the system is reflected in the naming of the topmost item in the model structure seen in the Figure 11. The part before “System Model” indicates the name of the system that the model represents, which in this thesis is the RRH.

Packages are the main way to organize a model repository. Their behaviour and purpose can be compared to operating system directories or folders we are all familiar with. The system model created in this thesis is organized into four main packages, which are: “base”, “domain”, “fcn”, and “log”. The “base” package is meant for storing model elements that, as the name suggests, define the base of the system. In this thesis it is used for requirements. The “domain” package is used for storing data items, i.e. the types of signals or data that the systems uses. The “fcn” package contains model elements used to describe to the functional architecture of the system. The “log” package contains elements that describe the logical architecture of the system. As can be seen in the Figure 11, the structure also contains inner packages, which are used to further organize the model. Their purpose will be discussed in detail in the later chapters.

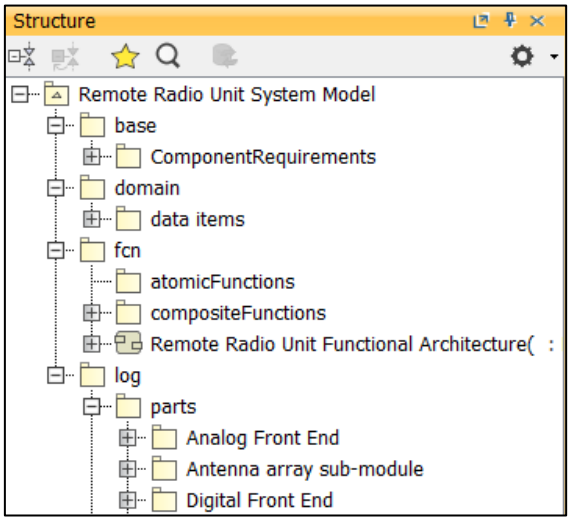

Figure 11. Package structure of the system model.

4.2 Modeling of Requirements

As mentioned in the background section of this thesis, information of a system's requirements is essential in system modeling. In SysML, requirements are modeled with "Requirement" elements and stored under the "base" package. These elements have three main properties, which are "Name", "ID", and "Text". In this example the name property is used for a short description of the requirement. The ID property is used for its identifier number defined in DOORS Next. The "Text" property is used for the detailed description of the requirement.

However, requirement management tools like the web-based DOORS Next, are principal software applications for storing and managing engineering requirements. For this reason, it is crucial to establish a connection between the requirements management tool and the system modeling tool. With a connection, requirements that match the ones in DOORS Next can be easily created in the system model. In addition to this, the connection makes sure that if changes in the requirements happen in DOORS Next, their counterparts in the system model can be automatically updated.

An additional plugin tool for Cameo called DataHub is used to establish the connection. In practice this starts with selecting the DOORS Next from the drop-down menu in the DataHub's GUI, which can be seen in Figure 12. After this, the server provider URL and user credentials for DOORS Next are given so that the tool can access the application.

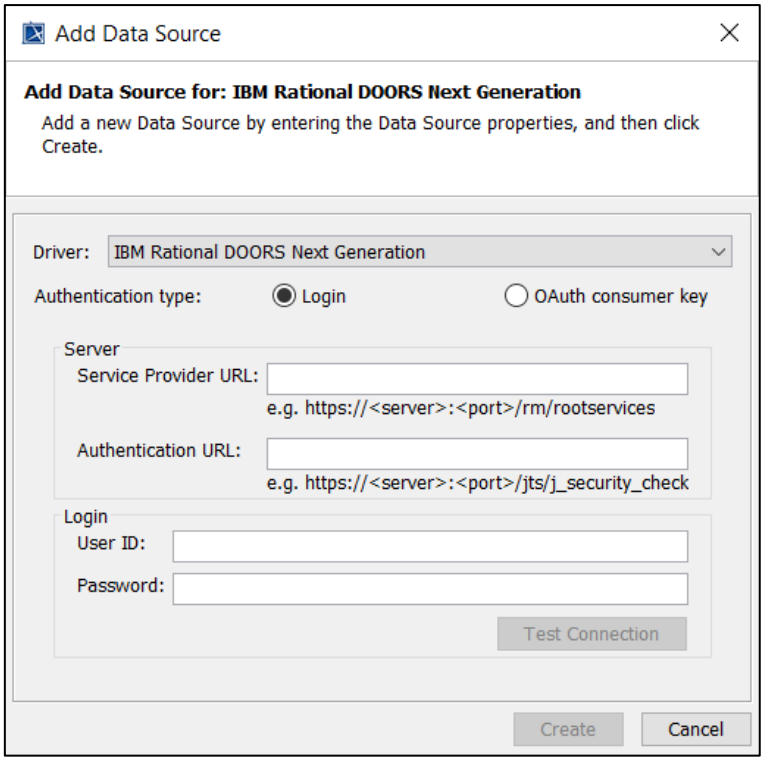

Figure 12. Connecting to DOORS Next with DataHub.

After a successful connection to DOORS Next has been made, its data structure becomes visible in a DataHub Explorer GUI. Now the user can navigate through the DOORS Next folders and locate the requirements of interest, which can be seen in the Figure 13.

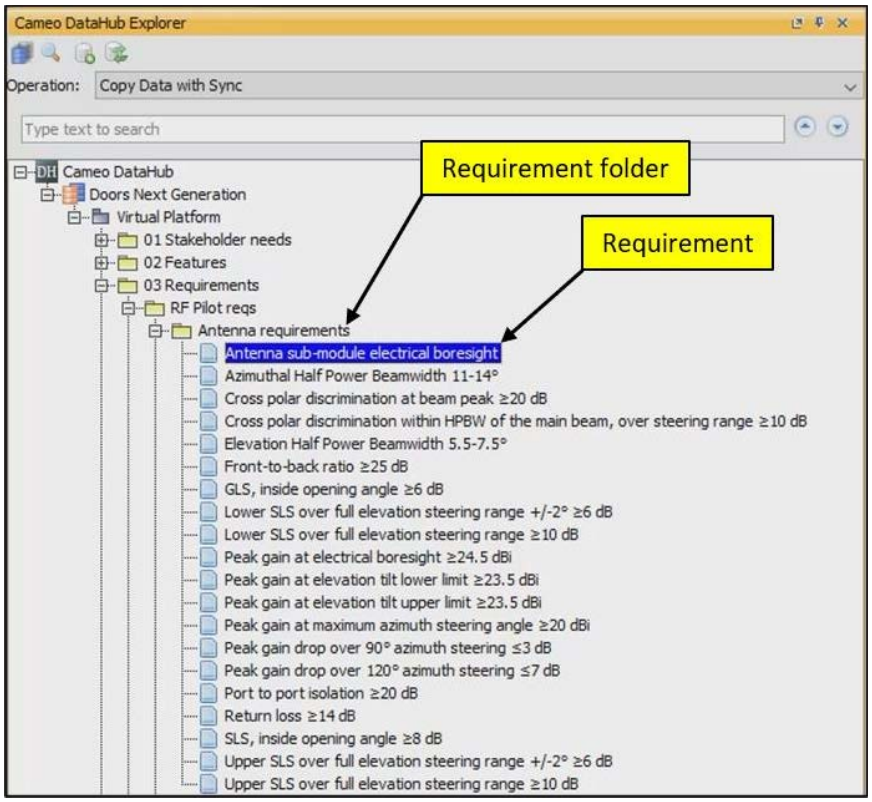

Figure 13. DOORS Next data structure accessed as seen in the DataHub Explorer GUI.

By simply dragging and dropping the requirement items from the DataHub Explorer to a desired location in the model repository of the system model, a “Copy Data with Sync” interface, seen in Figure 14, appears. Direction of the synchronization is chosen first. This configures the direction from which changes to the requirement can be made. For a demonstration purpose, two-way synchronization option has been chosen, which means that the requirement can be changed and updated from both DOORS Next and Cameo. In real product development however it is be essential to clearly define who is allowed to modify the requirements and through which software application.

The interface is also used to configure the SysML element type, which will be created in the system model-based on the item dragged from DataHub. Requirement element type is chosen from the list as seen in the Figure 14. Finally, the attributes of the requirement to be brought from DOORS Next, and the properties of the SysML requirement element into which they will be mapped, are all selected in the interface seen in the lower part of Figure 14.

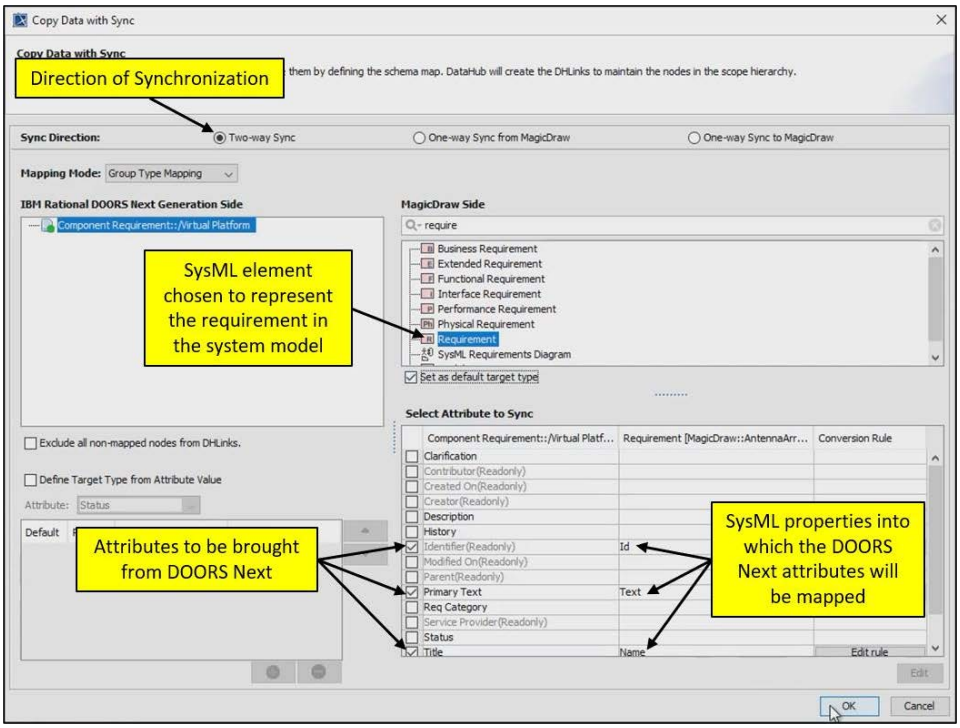

Figure 14. Copy data with sync interface.

With these configurations, a SysML requirement element synchronized with DOORS Next, and a hyperlink to the requirement in DOORS Next are created in the model repository for all the requirements listed in the chapter 3.2, as can be seen in the Figure 15. The antenna array requirements are placed under a package allocated just for them, so that the model stays well organized.

Finally, constraints are extracted from the requirements, which are mathematical expressions for the requirement, for example ≤ 5. Cameo is able to do this automatically if the requirements have been written unambiguously and with concise language. After this all the requirements have been modeled and they are ready for analysis and further use within the model.

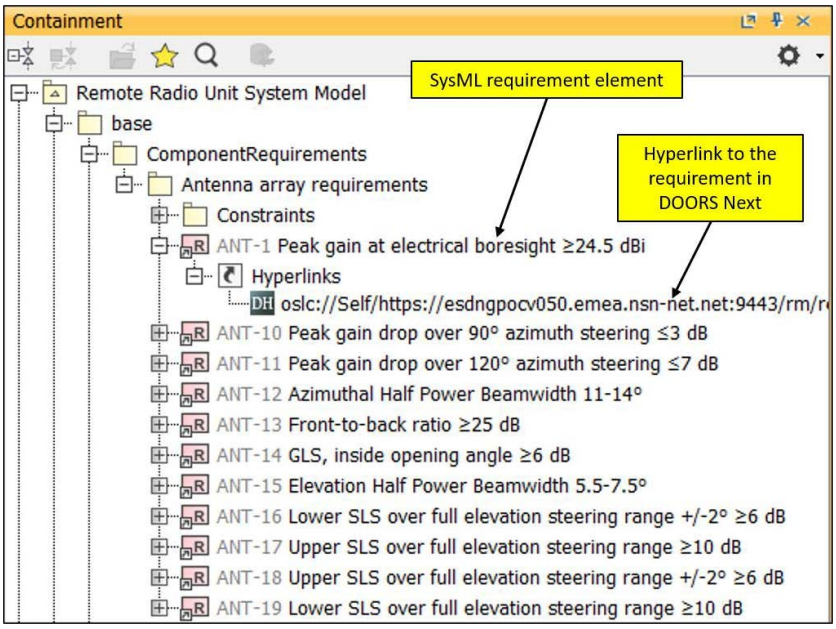

Figure 15. SysML Requirement elements in the model repository.

4.3 Modeling of Functional Architecture

In the MBSE methodology of Nokia, functional architecture has been defined as an implementation technology independent description of the functional aspects of a system. In a fully fledged system model, its main purpose is to act as a bridge between the functional requirements of a system and implementation of the system functions. Since this use case does not include any functional requirements, and the function of an antenna array is fairly straightforward, the functional architecture created is very simple and its purpose is to merely demonstrate this type of modeling.

Functional architecture of a system is built from flows, system functions, and interfaces. Flows can be information or physical items that flow from one function to another. A flow item going to a function is an input and an item coming out of a function is an output. Functions describe what happens to its inputs so that outputs are produced. Interfaces or pins are part of functions and specify their allowed inputs and outputs. The functional architecture is visualized in SysML activity diagrams.

System functions are modeled with SysML Activity elements and stored under the “fcn” package. The functions are further categorized as atomic or composite functions, which have their own designated packages in the model repository. Atomic functions are undividable singular functions and composite functions are functions that aggregate two or more functions. This helps in organizing large amounts of atomic functions.

System functions consist of actions that are either modeled with SysML call behaviour action or opaque action elements. Call behaviour action elements are used to model actions that can be further divided into multiple actions. Opaque action elements on the other hand are used for actions that cannot be divided any further. Like functions, these elements too have flows between them and pins that specify their inputs and outputs.

Since the system model in this use case is the RRH, the functionality of an antenna array can be thought as a part of a system function that contains all analog actions of a radio transceiver, spanning from the antenna to the digital and analog converters.

The modeling starts with creating an activity diagram named “Remote radio head Functional Architecture”, which is used to visualize the highest-level of the functional architecture of the system. All the system functions could be visualized and created in this diagram, but only one activity element named “Perform analog radio transceiver actions” is created. This element represents the system function mentioned earlier. In addition to this, four main interfaces are created for the system with SysML Activity Parameter Nodes, which include two inputs and two outputs. Finally, the flows between the interfaces and the function are created with the SysML object flow connectors. The resulting diagram can be seen in the Figure 16.

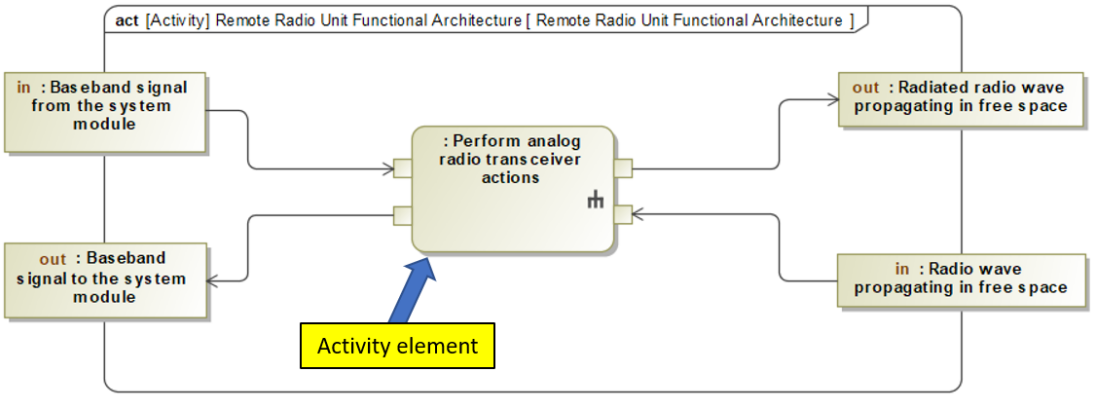

Figure 16. Remote radio head functional architecture activity diagram.

Another activity diagram is created inside the activity element, in which its internal behaviour can be modeled with different actions. Since a transceiver transmits and receives, the function needs two action flows in opposite directions.

Actions could now be created in the diagram to represent all actions of the transceiver, such as filtering, amplification, and digital and analog conversions, but in this use case, only two that correspond to the antenna array are created. Figure 17 shows the two actions that represent the functionality of the antenna array. The action on the left represents the transmitting action of the antenna with a flow to the function's output interface, and the action on the right represents the receiving action of the antenna with a flow that comes from the function's input interface. The same interfaces seen in the previous diagram are present here as well. All other transceiver functions have been simplified and modeled by two ideal actions "transmit" and "receive".

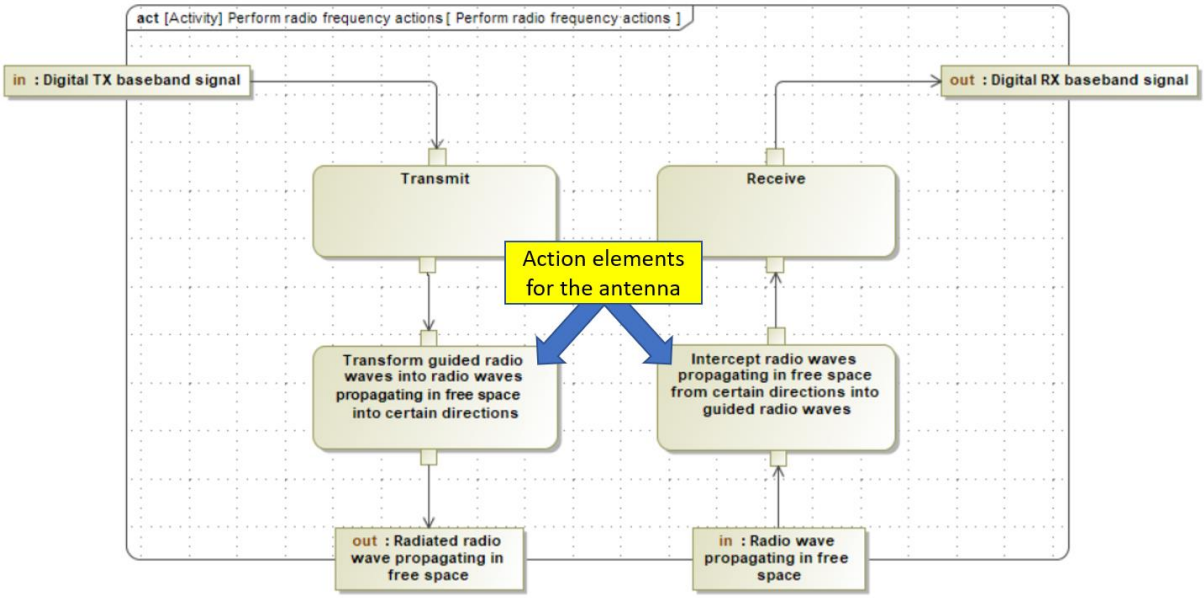

Figure 17. Antenna array functionality described with two action elements in an activity diagram.

4.4 Modeling of Logical Architecture

In the MBSE methodology of Nokia, logical architecture is defined as an implementation technology independent architecture of a system. It serves as the intermediate level of abstraction between requirements, functional architecture, and domain specific models in external tools. In it the system is divided into smaller and smaller subsystems or components, eventually reaching an undividable level from the system perspective. These subsystems and components are abstractions of the parts of the physical system, capturing their qualities and functions without imposing any implementation constraints.

Logical modeling happens in SysML block definition and internal block definition diagrams. The block definition diagram (BDD) is used to capture the system's or its component's logical composition. This view shows the compositional relationships between the system's subsystems and components, i.e. which part consists of which parts. The internal block definition diagram (IBD) is where the structure and the interconnections of a system or its components are modeled. Several IBDs are usually created to capture the structure of the system at different abstraction levels or to focus on the details of a single component.

The logical architecture and its parameters are modeled for the antenna array using the two diagrams mentioned. First a BDD diagram is created under the package "log" in a package specifically designated for the antenna array component. The logical composition of the antenna array can now be modeled in the diagram.

As previously discussed in the chapter 3, the antenna consists of 32 sub-arrays and from two feedline networks, which for simplification purposes are considered as one. Furthermore, it was mentioned how each sub-array consists of two array elements, one per polarization, each of whom consists of three radiating elements. Using this information, the logical composition of the antenna array can be formed. The antenna array and its each logical part are modeled with a SysML block element and connected to each other with SysML directed composition associations. The resulting diagram can be seen in Figure 18. The topmost element is the antenna array itself and the lower elements show its hierarchical composition. There is also a number next to the sub-array block that indicates its multiplicity, i.e. how many are in the structure. The multiplicity of the radiating elements is modeled with multiple directed composition associations from the array element blocks to the same radiating element block, this can be seen in the lower part of Figure 18.

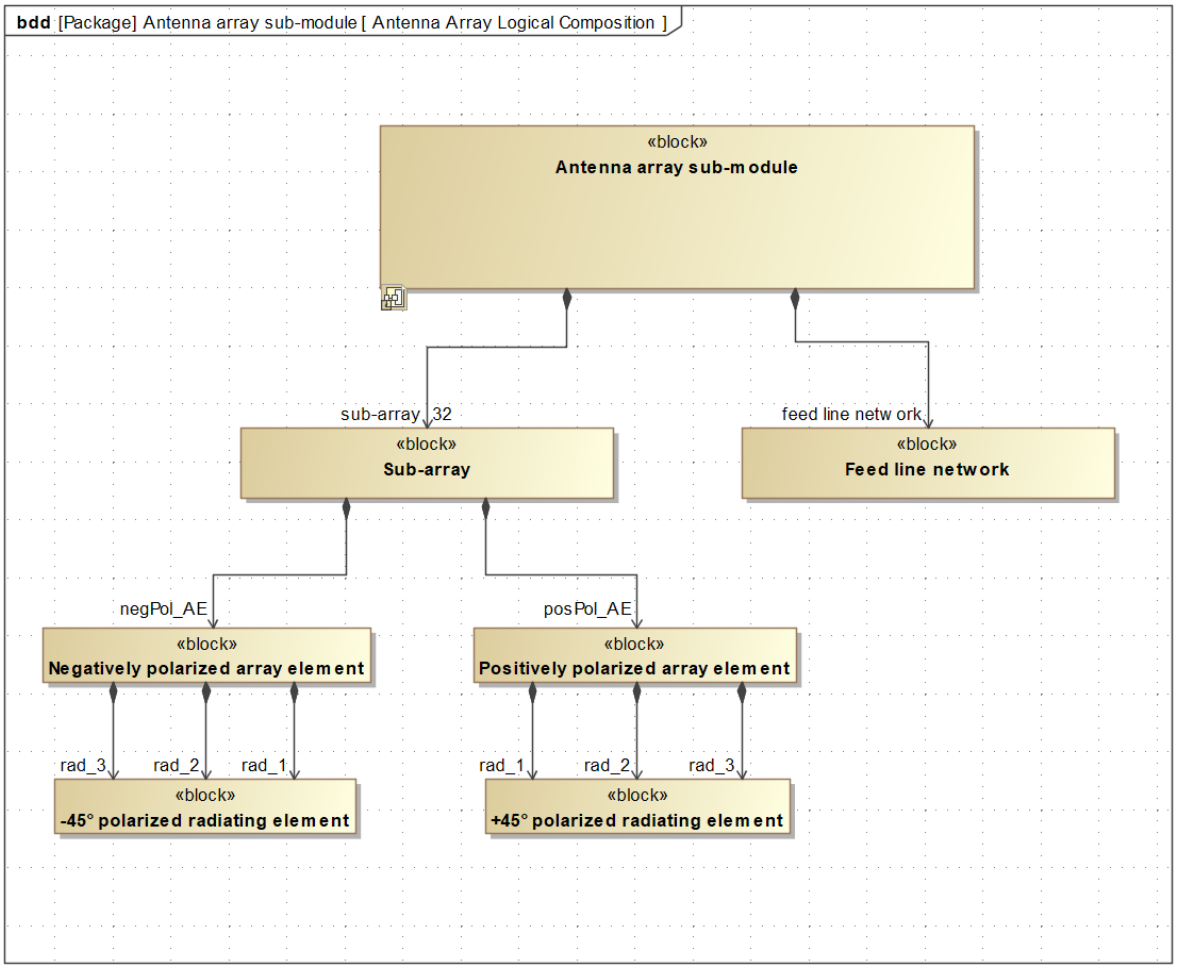

Figure 18. Antenna array logical composition in a block definition diagram.

Next, an IBD diagram is created for the antenna array sub-module block. Like previously mentioned, the block and its parts and their ports and interconnections are displayed and modeled in this diagram. First the logical ports of the antenna array are modeled with SysML proxy port elements. One is added to left edge of the diagram and another to the right edge. The left port represents the antenna ports that are connected further into the 64 transceivers. The needed multiplicity is achieved with the configuration of the port's multiplicity parameter. The right port represents the antenna's connection to the surrounding radio environment.

Proxy port elements are added also to the parts of the antenna array. On its transceiver side, the feed line network has 64 ports, matching the ports of the antenna array. On the side facing the radiating elements, the feed line network has 32 ports for one polarization and 32 for the another. The sub-array and array elements both have one port, enabling the connections between the feed line network and the radiating elements. Following this, two ports are added to the radiating elements, one to connect to the feedline network and another to connect to the port that represents the natural connection of the antenna to the surrounding radio environment. Afterwards, connections between the ports are modeled with SysML connectors. Finally, item flows are added to the connectors between the feed line network and the sub-array. This highlights the type and the direction of the signals flowing through. Since this exercise is done with a transceiving antenna the same type of signal flows in both directions. The final diagram resulting from these modeling actions can be seen in the Figure 19.

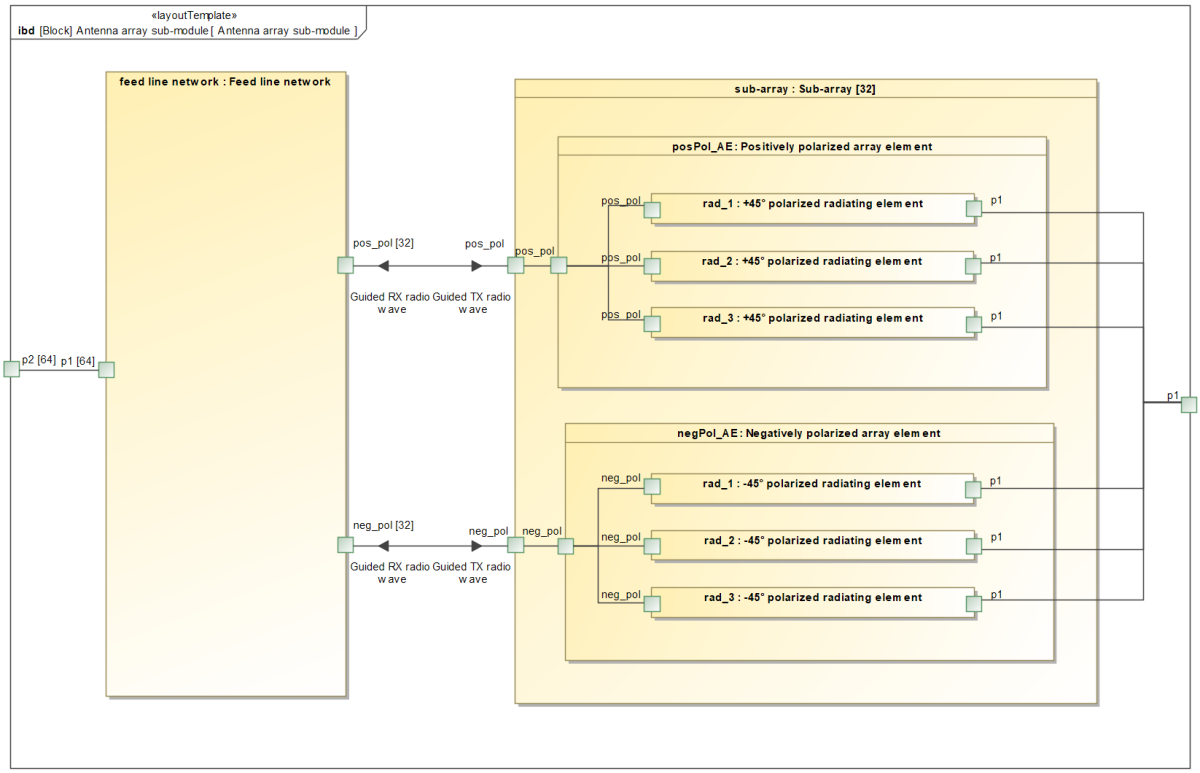

Figure 19. Antenna array logical structure in an internal block definition diagram.

The final step that concludes the modeling of the logical architecture of the antenna array is its parametrization and connecting them to the previously modeled requirements. Another BDD diagram is created for this purpose. In this diagram, design variables and performance evaluation parameters are created for the antenna array. The performance evaluation parameters are used to determine whether the design of the antenna array satisfies the requirements, therefore they have to be connected to the requirements.

The antenna array sub-module block and the requirement elements are dragged and dropped to the diagram from the model repository where they were created earlier. After this, SysML value properties are created for the antenna array sub-module block. They are given descriptive names and correct value types and units. As an example, one value property is named “peakGainBoresight”, its type is “ratio to isotropic radiator”, and its unit is defined as “dBi” This and all the other value properties represent the performance parameters of the antenna. After all the parameters have been defined, they are connected to their corresponding requirements with SysML “satisfy” relationships. Since all parameters and requirements cannot fit into one picture, only two are modeled and connected in the diagram seen in Figure 20. The two value properties can be seen on the left inside the block, and the corresponding requirements on the right.

This concludes the system modeling of the antenna array. It can be now used as the central hub for an integrated model framework. In the next chapters, an analysis model for the antenna array is created, which is then integrated to the system model with the MDAO tool ModelCenter. This will show the integrated model framework in action with analysis.

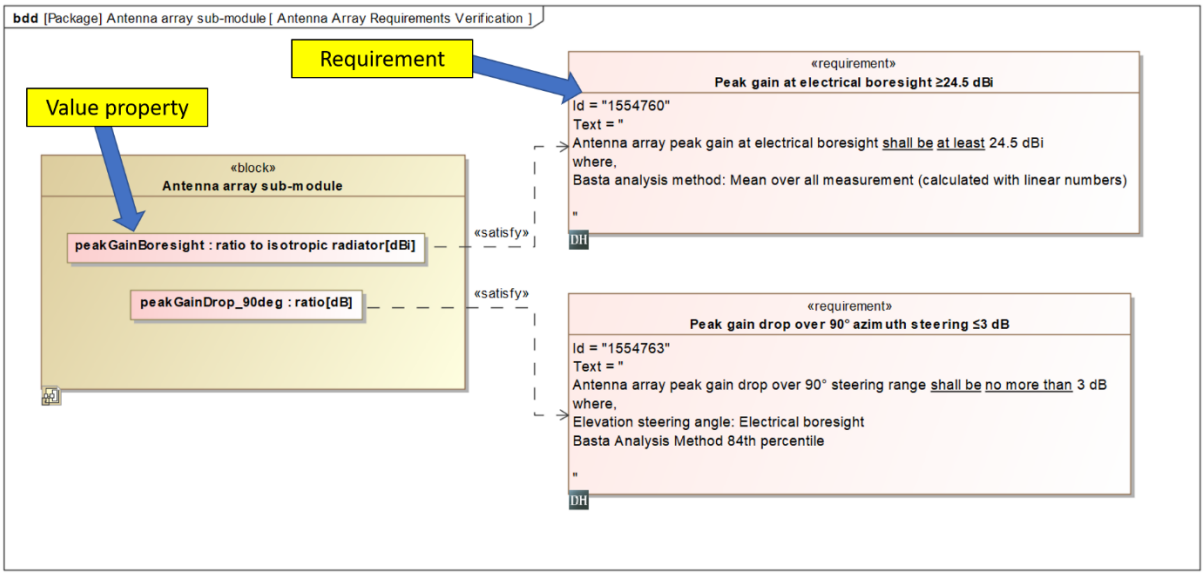

Figure 20. Antenna array parameters and corresponding requirements connected in a block definition diagram.

5 ANTENNA ARRAY ANALYSIS MODELING

The next part in this use case is the creation of analysis models for the antenna array at three different fidelity levels, with the modification capability of the spacings of the array. The term fidelity is used in this context to describe how accurately the model represents its real-world counterpart. High fidelity models are naturally the most accurate but computationally expensive and therefore, require a lot of time to solve. Low-fidelity models on the other hand, offer less accuracy but are much faster to simulate. Three different fidelity level models are created in this use case. These models are all built from two different components, which are an electromagnetic simulation model created in Ansys HFSS, and an array analyser and data post-processing script created in MATLAB.

The larger feed line network that is responsible for creating the sub-array structure, is omitted from all these models because its design depends on the spacings of the array. Instead, the sub-array implementation is done with ideal mathematical functions that divide the signals into sub-arrays. The smaller feed line network, however, is included in all the models since it does not depend on the spacings.

5.1 Low-Fidelity Model

Lowering the fidelity of a model is done by leaving details out of the model that are computationally intensive and whose neglection will not significantly worsen the accuracy of the simulation results. If the models are constructed in this way, despite their lower accuracy, they can be used together with DOEs to find correct correlational relationships between design parameters and performance parameters in much less time than with higher fidelity models. In addition to this, they can be used to find out design configurations that are not feasible i.e. do not satisfy the requirements. With these results from the low-fidelity model, the design space can be narrowed down, meaning that the amount of possible designs can be minimized before moving on to analysis with higher fidelity models that take a much longer to simulate. Let us consider next why and how a low-fidelity model for the antenna array is created.

The antenna array in this use case is large, consisting of 12 by 8 radiating elements. Simulating its total radiation pattern in Ansys HFSS is computationally heavy. This is because HFSS does it by first, solving the electric and magnetic fields of each radiating element, then calculating their sum, and finally calculating the radiation pattern from the total electromagnetic field. Therefore, time can be saved if the number of radiating elements in the model are reduced.

The total radiation pattern of a smaller array is of no use in the analysis of a larger array. However, the radiation pattern of a single radiator could be used together with the array factor to form the total radiation pattern of an array of any size. Therefore, it would seem that the simplest option would be to just model a single radiator and solve its radiation pattern. However, this approach would completely omit the effects the surrounding elements have on the radiation pattern of a single element in an array. For this reason, a model has to be created in HFSS, in which the radiation pattern of single element can be solved while it is surrounded by other radiators.

A 3-by-3 array is the smallest possible rectangular array with an element surrounded by other elements symmetrically. This model's centre element's radiation pattern was found to be a good estimate for the radiation pattern of individual elements in the full 12-by-8 array, especially when it comes to the single element's maximum gain and HPBW.

The radiation pattern of an ideal radiating element called cosine element follows a cosine function raised to a certain power in both the azimuth and elevation directions. By modifying

the exponents, the HPBW of the pattern can be changed in both directions. In order to combine this ideal model with the electromagnetic simulation results, the exponents are calculated so that the resulting HPBWs of the ideal radiation patterns match the ones obtained from HFSS. In addition to this, the gain of the cosine element is matched by simply summing the maximum gain value obtained from HFSS to its radiation pattern. These actions result in a cosine element that is in a way calibrated based on the electromagnetic simulation results.

Based on this information a 3-by-3 array model is constructed in HFSS from the radiating elements introduced in the chapter 3.1. The resulting model can be seen in Figure 21. Furthermore, azimuth and elevation spacing design parameters are added, so that by simply changing their values, the model changes accordingly. Finally, the solution options of the model are configured. The fields of the model are set to be solved at five discreet frequency points at regular intervals, which are 3.4 GHz, 3.55 GHz, 3.7 GHz, 3.85 GHz, and 4.0 GHz.

After the electromagnetic fields of the model have been solved, a plot containing the radiation pattern of the centre element in 3D spherical coordinates at all the solved frequencies is created. The data in this plot is exported in a .csv file format, which is fitting for further analysis in MATLAB.

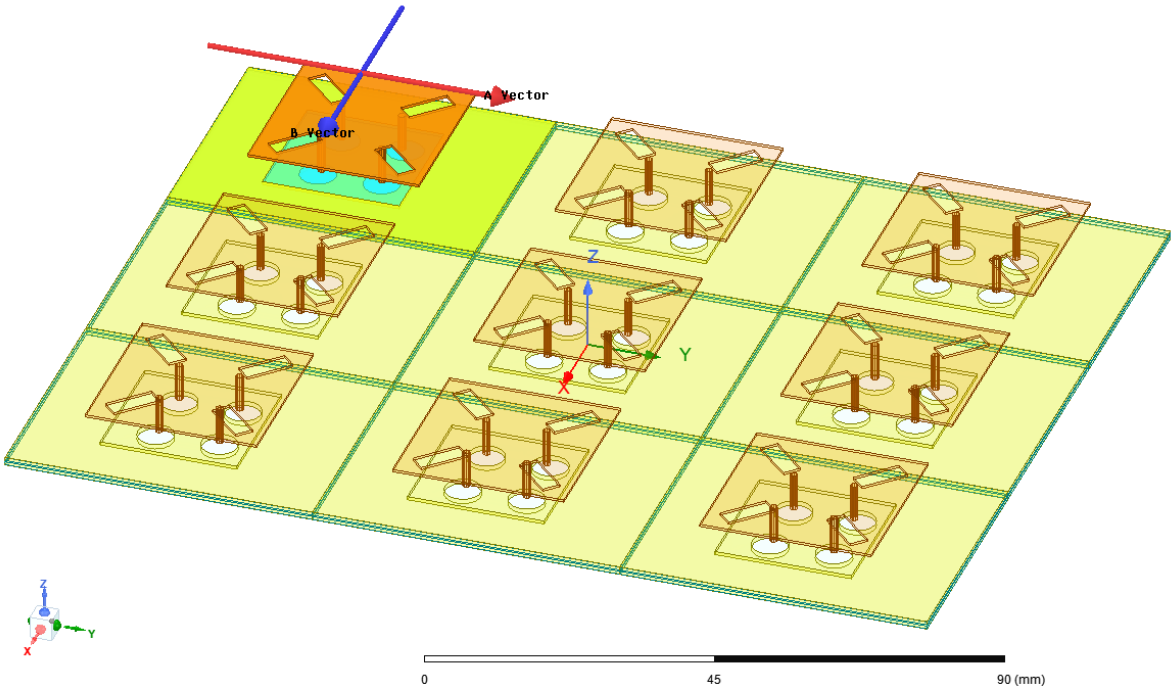

Figure 21. Low-fidelity analysis model in Ansys HFSS.

In order to complete the analysis, a MATLAB script is created that is able to read the radiation pattern data exported in .csv format from HFSS. The script calculates the gain and the elevation and azimuth HPBWs of the HFSS element from the exported data. Subsequently the script forms an ideal cosine radiation pattern whose gain and HPBWs matches the radiation pattern data from HFSS, so that it is in a way calibrated by the HFSS model. Afterwards the script calculates an array factor based on the array configuration given to it as inputs listed in the Table 3. After this, the script combines the calibrated cosine element's radiation pattern and the calculated array factor, producing a total radiation pattern for an array. This is repeated at all the five frequencies, and at all the needed beam steering angles.

This is done by calculating the ideal phases needed for each element in the array in order to steer the main beam to a certain angle in 3D space. These phases are then used to calculate new array factors from which new ideally beam steered radiation patterns are formed.

Finally, performance parameters matching the requirements listed earlier in the chapter 3.2, except FBR, XPD, isolation, and return loss, are all measured from the radiation patterns. Their final values are formed with their respective BASTA analysis methods. These performance parameters are the output of the script, and they can be directly used to evaluate whether the antenna array design meets its requirements.

Table 3. Input variables in the low-fidelity model

Input name	**Description**	**Unit**
numOfRows	Number of vertical radiating elements	-
numOfColumns	Number of horizontal radiating elements	-
verticalRadElemSpacing	Spacing between vertical radiating elements	Millimetre (mm)
horizontalRadElemSpacing	Spacing between horizontal radiating elements	Millimetre (mm)
f_min	Lowest simulated frequency	Gigahertz (GHz)
f_max	Highest simulated frequency	Gigahertz (GHz)
numOfFreqs	Number of simulated frequencies	-

5.2 Medium-Fidelity Model

First way to increase the fidelity from the low-fidelity model is to leave out the usage of the cosine element. The accuracy of the radiation pattern of a single element in HFSS has to be first improved from the low-fidelity model before it makes sense to use it directly in MATLAB. This is because the small size of the low-fidelity model does not model FBR accurately, and since it only uses the centre element, the differences between the radiation patterns of radiators resulting from their positioning are not taken into account.

First step is to enlarge PCB. This is because the size of the PCB determines how much the array radiates backwards, measured with the front-to-back (FBR) parameter. Thus, to produce accurate results, the PCB's size must match the size it would have in the real 12 by 8 antenna array. This size expansion was found to increase the simulation time by just a couple of minutes thus making it an excellent way of increasing fidelity. In the previous model, FBR could not be evaluated since cosine radiators do not radiate backwards at all.

Second step is to increase the number of radiators. This makes the radiation patterns more akin to what they would be in the full 12 by 8 array, and instead of taking the radiation pattern of the center element, a radiation pattern is taken from all the elements and then an average radiation pattern is calculated from them. This averaged radiation pattern was found to produce very accurate results for the full radiation pattern of a 12 by 8 array when used together with the array factor calculation in MATLAB.

These findings resulted in a 4 by 4 array model in HFSS, constructed from the same radiating elements used in the earlier model. This medium-fidelity model can be seen in the Figure 22. Like the low-fidelity model this model too was configured to include the azimuth and elevation

spacing design parameters, and to be solved at the same five frequencies. But differing from the previous model is that now the radiation patterns of all the individual radiators are plotted and exported in the .csv file format.

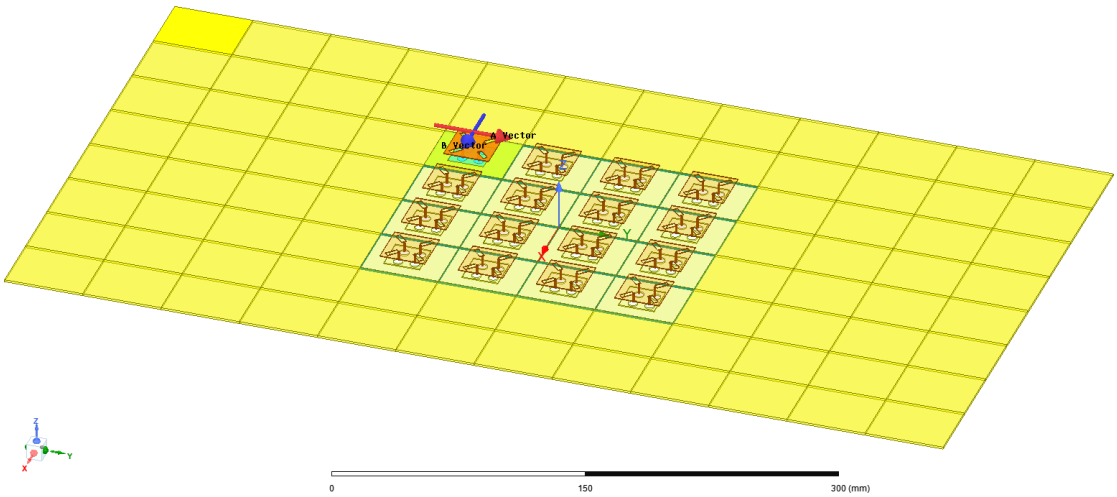

Figure 22. Medium-fidelity model in Ansys HFSS.

The MATLAB script used in this medium-fidelity model differs from the previous script in the way that now it calculates an average radiation pattern from all the patterns exported from HFSS. This average pattern is then used together with array factors to form full 12 by 8 array radiation patterns at all the simulated frequencies and at all needed beam steering angles.

The performance parameter evaluation part is almost identical to the script used in the low-fidelity model, but it now also includes the calculation of the FBR. The inputs of the model are identical to the ones in the low-fidelity model.

5.3 High-Fidelity Model

In the high-fidelity model, the aim is to include as many details of the antenna array as possible. This means that the full 12 by 8 antenna array is modeled in HFSS. Consequently, the full radiation pattern is no longer solved in MATLAB but directly in HFSS at all the five frequencies and even steered to all the needed angles.

More performance parameters can be evaluated in the high-fidelity model. These are cross-polar discrimination, isolation between radiating elements and return loss. Since these parameters cannot be derived from radiation patterns, additional plots are needed.

Determination of XPD requires a plot of the ratio of the co-polar component of one polarization compared to orthogonal cross-polar component at all the simulated frequencies. Determination of isolation on the other hand requires plots of the s-parameters between the ports of the same polarization of two elements located next to each other in the middle of the array and of two located next to each other in a corner of the array. And finally, the determination of return loss requires plots of the S_{11} parameter of the two ports of radiators located in the middle of the array and of two located in a corner of the array. Radiators both in the middle and in a corner are considered so that the effect of relative positioning is taken into account. Isolation is worse between the radiators located in the middle of the array and return loss, on the other hand, is worse for radiators located on the edge of the radiator. All these plots are exported in the .csv file format.

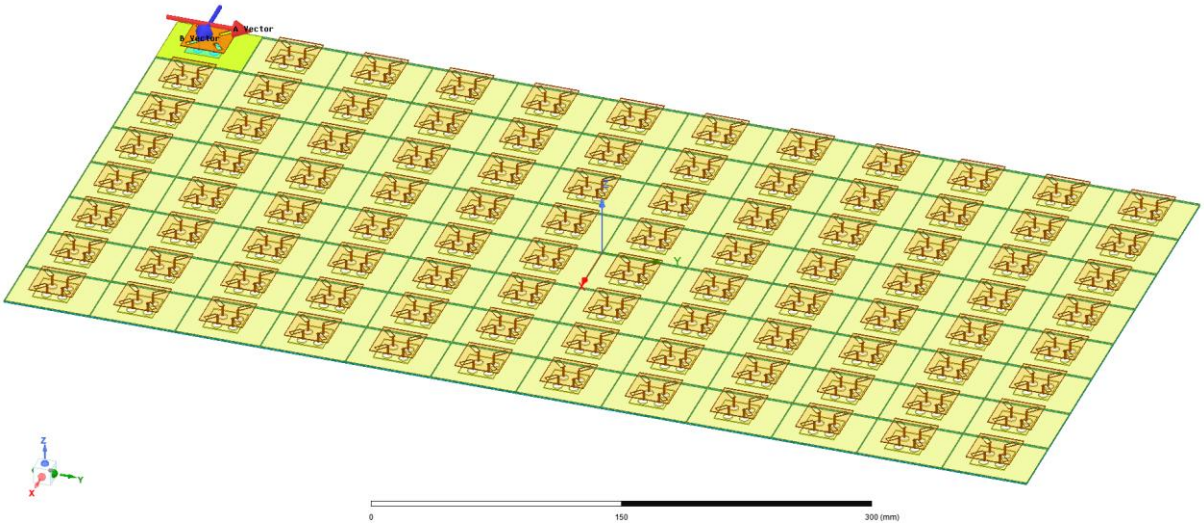

Figure 23. High fidelity model in Ansys HFSS.

Since the radiation pattern data exported from HFSS is already for the full array and also contains the needed beam steered radiation patterns, the MATLAB script no longer needs array factor calculation and can straight away start with the performance parameter evaluation. This is almost identical to the evaluation done in the previous scripts except that now also the XPD, isolation and return loss data, are all analysed and needed parameters are formed with their respective BASTA analysis methods listed in the chapter 3.2. The inputs of the high-fidelity model are exactly the same as in the previous two models.

5.4 Comparisons Between the Models

Since each model has its own strengths and weaknesses, comparing them to each other is important. The Table 4 lists the performance parameters each model has as its outputs, therefore is able to evaluate.

Table 4. Comparison between different fidelity models for the antenna array

Performance parameter	**Low-fidelity Model**	**Medium-fidelity model**	**High-fidelity model**
Peak gain at electrical boresight	yes	yes	yes
Peak gain drop over 90° steering range	yes	yes	yes
Peak gain drop over 120° steering range	yes	yes	yes
Azimuth HPBW at electrical boresight	yes	yes	yes
FBR over azimuth and elevation steering range	no	yes	yes
GLS inside 180° azimuth opening angle,	yes, but not very accurate	yes	yes

over 120° azimuth steering range			
XPD at beam peak, at electrical boresight	no	no	yes
XPD within HPBW of the main beam and over azimuth and elevation steering range	no	no	yes
Elevation HPBW at electrical boresight	yes	yes	yes
Upper and lower elevation SLS, over full elevation steering range	yes, but not very accurate	yes	yes
Upper and lower elevation SLS, over full elevation steering range +/-2°	yes, but not very accurate	yes	yes
Isolation	no	no	yes
Return loss	no	no	yes

6 ANTENNA ARRAY ANALYSIS MODEL AUTOMATION AND INTEGRATION

In this part of the use case, HFSS and MATLAB are automated in ModelCenter, and integrated into a complete analysis workflow. Finally, the workflow is connected to the system model created earlier in Cameo Systems Modeller. This is achieved in four steps that are presented in the Figure 24. Due to time limitations, these steps are only demonstrated for the low-fidelity analysis model. For this reason, the analysis results achieved with this model can only be considered as coarse approximations. In reality, the analysis would continue with the higher fidelity level models that would produce more accurate results.

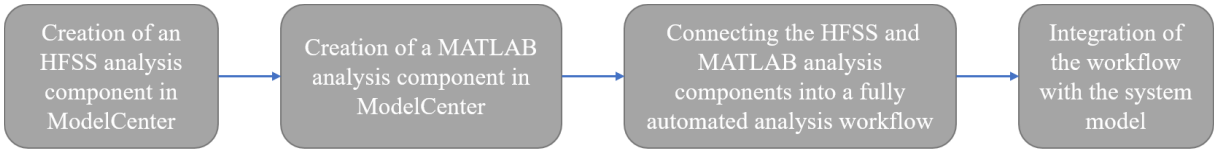

Figure 24. Steps in the automation and integration of the antenna array analysis model.

6.1 Automation of HFSS and MATLAB

Like mentioned in the background section of this thesis, advanced tools like HFSS require Python scripting in order to be automated in a ModelCenter workflow. Ansys HFSS has an in-built Python script recorder. With it, any actions the user makes in the application, are recorded into commands in a python script file. After the recording has been finished, the resulting script can be executed, automatically completing all the recorded actions.

For the purpose of the antenna array use case, a script is recorded for the low-fidelity HFSS model, which is able to modify its simulation frequencies and its design parameters, create the needed plot for the radiation pattern of the centre elements, initiate the simulation, and finally export the radiation pattern data in a .csv file. As a result, this script automates everything required for analysis inside HFSS.

However, the execution of this python script requires HFSS to be already running. To automate this, a Windows batch command is needed. Any application can be started in Windows by simply calling its .exe file in the command prompt. For HFSS it is "*C:\\(full path to the HFSS installation directory)\ansysedt.exe*". Additional commands can be added on top of the execution. "-ng" launches the application in a non-graphical mode, meaning that the application is launched without any graphical user interface. "-runscriptandexit" -command automatically runs a specified script in HFSS and closes the application after the script has been completed. Combining all these commands into one results in "*C:\\(full path to the HFSS installation directory)\ansysedt.exe -ng -runscriptandexit scriptName.py*". The execution of this batch command can be implemented in Python with the subprocess module. As a result, two Python script are created in total to automate analysis in HFSS, the batch command script, and the HFSS model modification script.

These two scripts are integrated to a ModelCenter workflow with its QuickWrap feature. In the QuickWrap GUI, the HFSS model modification script is specified as an input file and the running of the execution script is given as its run command. After this, ModelCenter automatically recognises all variables in the script. The script contains six variables that are chosen to be imported to ModelCenter as inputs. These are as follows, the name of the model

(fileName), azimuth spacing (horizontalRadElemSpacing), elevation spacing (verticalRadElemSpacing), the number of simulated frequencies (numOfFreqs), lowest simulation frequency (f_min), and the highest simulated frequency (f_max). The resulting view in the QuickWrap GUI can be seen in the Figure 25, in which the input variables, and the run command, are both highlighted.

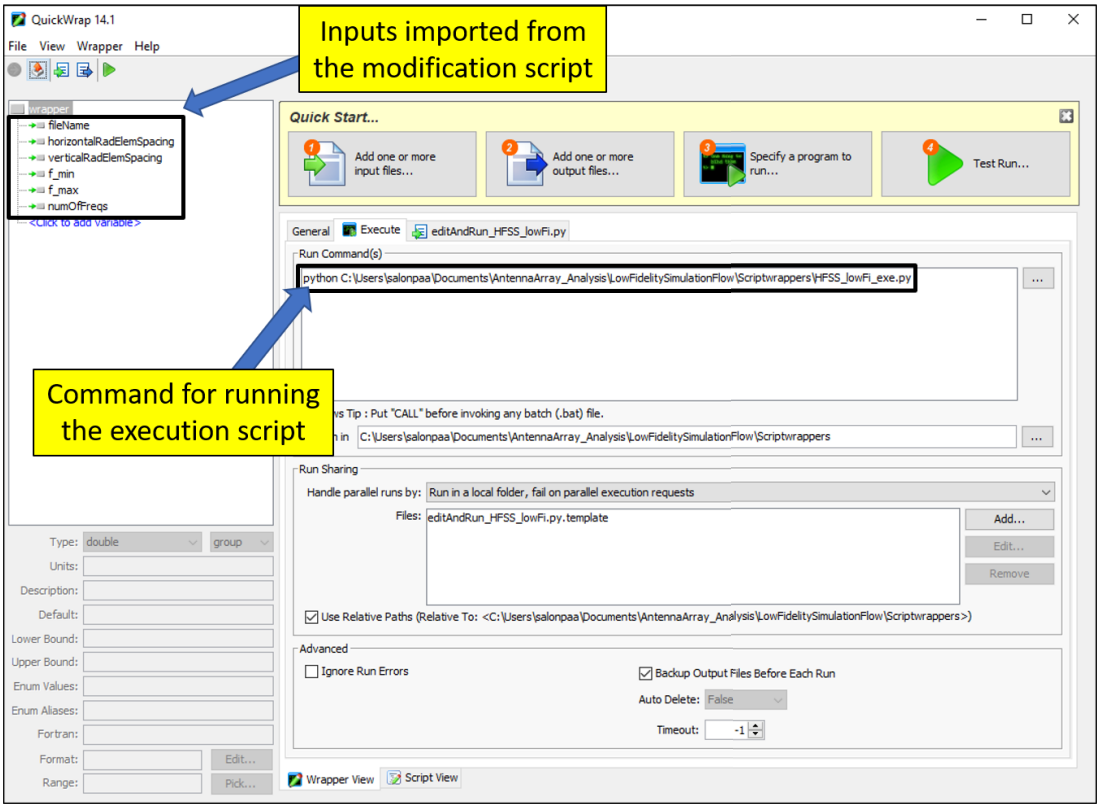

Figure 25. Automation of the low-fidelity HFSS model with QuickWrap in ModelCenter.

These actions result in the first component in the analysis workflow, a QuickWrap component that is able to run HFSS, modify the low-fidelity model with the parameters specified, simulate it, and export the resulting radiation pattern in .csv format. Now the user is able to change any of the parameters and run the component. However, the workflow doesn't have any outputs. The MATLAB script that produces outputs from the radiation pattern file is still needed in the workflow.

Like previously mentioned, ModelCenter has an in-built plugin for MATLAB, but before a MATLAB script can be integrated, it needs a ModelCenter header in which all its inputs and outputs are specified. This includes the name, type, default value and description of each parameter. When such a header is added to the low-fidelity MATLAB script and then selected in the MATLAB plug-in GUI, all the parameters get automatically imported to ModelCenter as can be seen in the Figure 26. Now the workflow contains two components, the QuickWrap component that runs the low-fidelity HFSS model, and a MATLAB component that runs the low-fidelity post processing script.

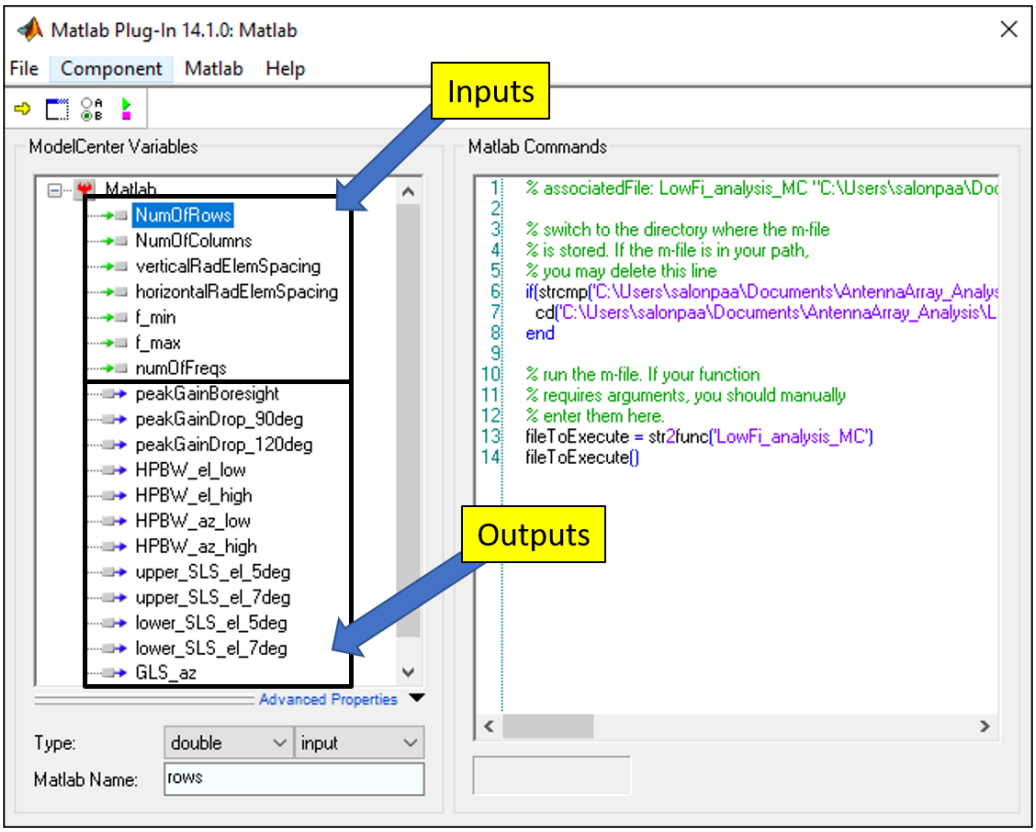

Figure 26. Automation of the low-fidelity MATLAB script with the MATLAB plug-in in ModelCenter.

As can be seen in the Figure 25 and Figure 26, the MATLAB component shares some of the inputs with the HFSS component. These inputs are connected to each other with the link editor of ModelCenter so that the values given to those inputs in the HFSS component get automatically copied to the MATLAB component. The model is now fully automated in ModelCenter and can be seen in the Figure 27. In practice, this means that the design variables or inputs can be modified, the workflow executed, and its results are captured as outputs. Next step is to integrate the workflow with the system model in Cameo.

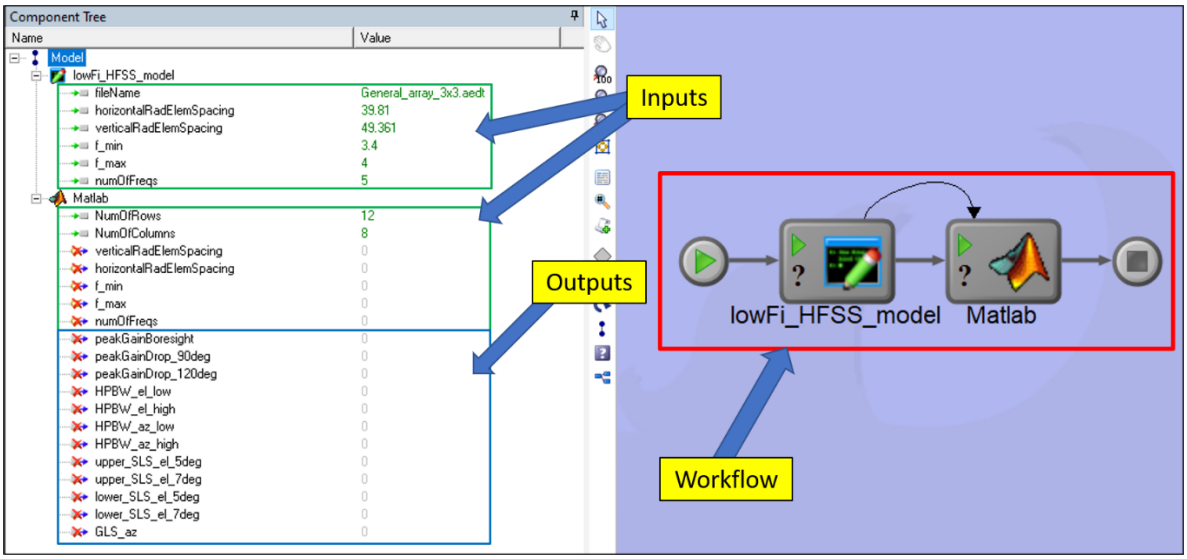

Figure 27. The complete low-fidelity antenna array analysis workflow in ModelCenter.

6.2 Analysis Workflow and System Model Integration

Integrating the workflow with the system model starts with packaging the workflow. This means that everything in the workflow is compressed into a single analysis component. In the packaging menu, seen in the Figure 28, different files needed to run the model can be chosen to be included in the package. In this example, only the workflow is included since its components are configured to use their respective files in their original locations. In addition to this, the parameters of the packaged workflow can be configured. This makes it possible to decide whether certain parameters are to be included in the packaged workflow or not, and to rename them. This is especially beneficial if, like it is in this use case, the original workflow includes duplicate input parameters or parameters that do not have system level significance. These are excluded from the packaged workflow. After these configurations are done, the package is created and stored to a predefined location on the PC.

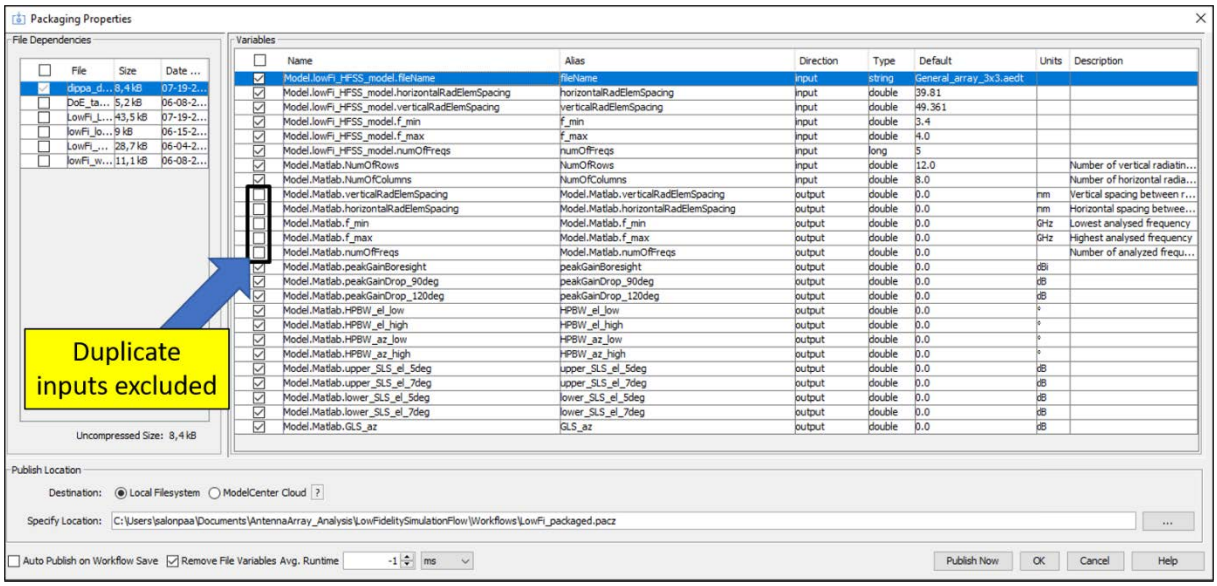

Figure 28. Packaging the workflow in ModelCenter.

Before the workflow can be integrated with the system model, a feature of ModelCenter called analysis server has to be configured. With it a certain location or folder on the computer is used to create a server. Other computers with ModelCenter installed and internet access are able to connect to this server and access the ModelCenter workflows or analysis components stored in the folder defined on the host computer. However, analysis server is not only used to share workflows and analysis components between different computers, but also as a way to enable the modularity and reusability of components on the same computer they are also physically located. Therefore, analysis server is started and the location into which the packaged workflow was saved is configured for the analysis server.

Now that the workflow has been packaged and the analysis server configured, the integration itself can be started in a ModelCenter MBSE Cameo plug-in. In this tool, an execution plan is created, which is a plan for a new analysis workflow that will be connected to the structure of the system model, its parameters, and requirements.

The construction of the execution plan starts with connecting the workflow to the parameters in the system model. This is done in the analysis editor of the ModelCenter plug-in. A connection is first made to the local analysis server where the packaged low-fidelity workflow is located. When it is chosen, its variables appear on GUI, seen on the right side of the Figure

29. The structure of the system model on the other hand is visible on the left. The structure is further navigated to the location with the SysML block element that represents the antenna array.

By dragging and dropping system model parameters under the antenna array block to their analysis counterparts, a link between them is established. A view of the analysis editor GUI after all the parameters have been linked is given in the Figure 29. By pressing OK, the connected workflow gets added to the analyses list of the ModelCenter MBSE plug-in. Finally, it is added to the execution plan by dragging and dropping it to the analyses section of the execution plan.

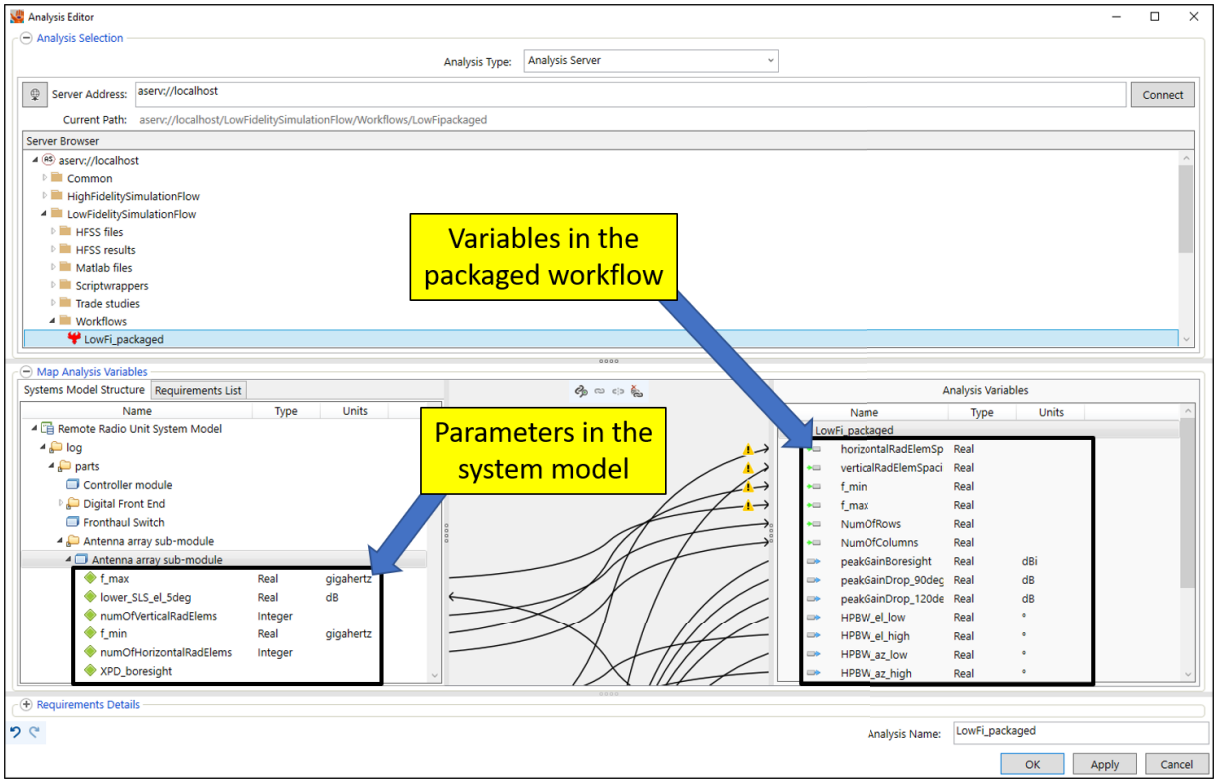

Figure 29. Connecting the packaged workflow with the system model.

Next task is to import the requirements from the system model to the execution plan. By simply selecting “import requirements” in the ModelCenter MBSE GUI, new analyses are created in the analysis list. Since constraints were created for each requirement earlier, ModelCenter is able to automatically create scripts for each requirement that analyse their conformance. However, the automatic script generation does not work on the HPBW requirements in this use case. This is because these are double sided requirements, meaning that their values are required to be in a certain range. A custom script is created with the Java programming language for the analysis of these requirements.

After all the requirements have their analysis scripts in the analyses list, they are added to the execution plan by dragging and dropping them to the analyses section of the execution plan. This finishes the construction of the execution plan and it is finally saved. A view of the completed execution plan setup with its different compartments highlighted is given in the Figure 30.

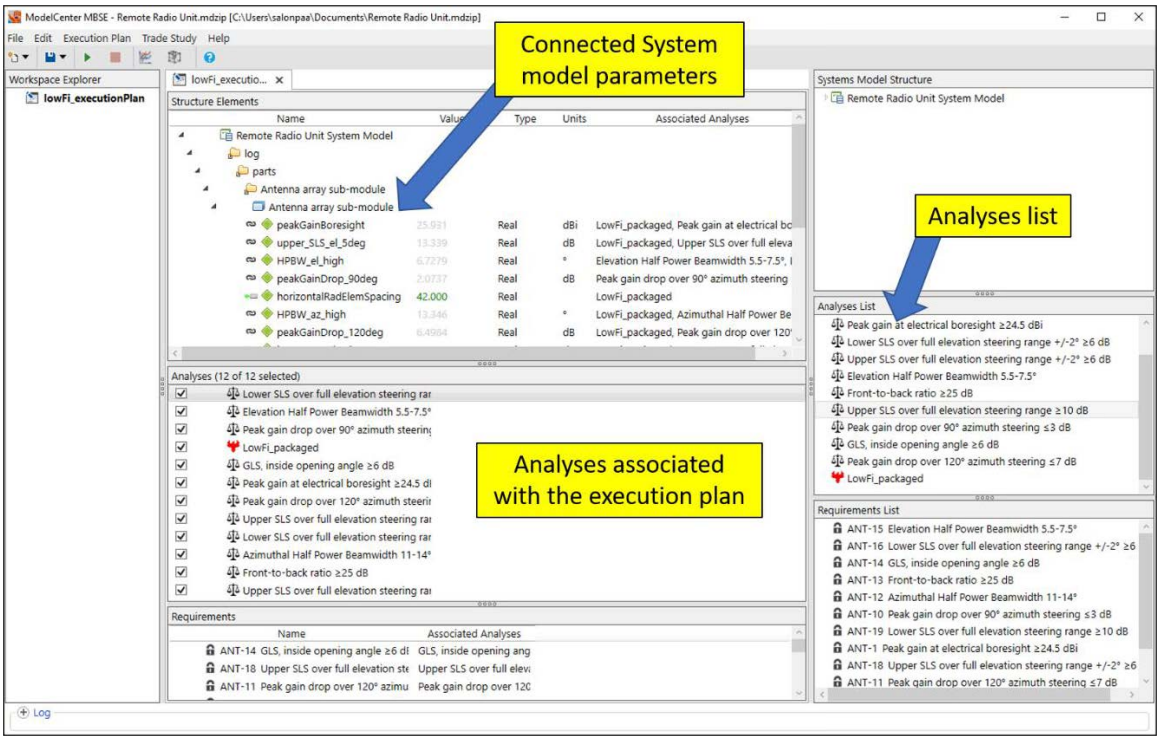

Figure 30. Completed execution plan in ModelCenter MBSE.

In order to perform optimization with a connection to the system model, the execution plan has to be turned into an actual workflow in ModelCenter. This is done by first, creating an empty workflow, and then opening the ModelCenter MBSE plug-in through ModelCenter while the system model is open in Cameo. This prompts a menu in which the antenna array model, currently open in Cameo, is selected. Selecting the model results in a GUI view that has on its left side the execution plan created previously, and when it is selected, the configuration of the plan appears on the right. Now the execution plan can be added into the empty workflow by selecting “Add to Workflow” in the upper part of the GUI, seen in the Figure 31.

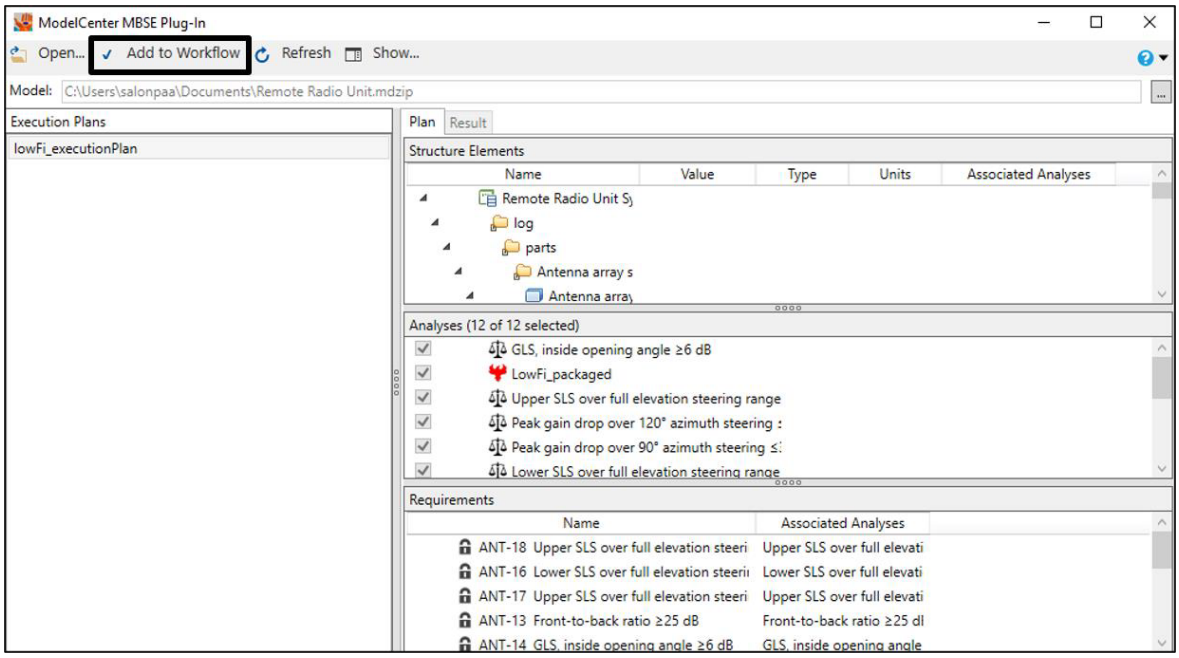

Figure 31. Adding the execution plan to a workflow.

This results in a workflow that can be seen in the Figure 32. The inputs and outputs of the workflow are fully integrated to the structure of the system model. Furthermore, the workflow includes the requirements from the model and analysis scripts that verify them. The next chapter will discuss how DOE and optimization are performed for the antenna array using this connected workflow, thereby going further into its details.

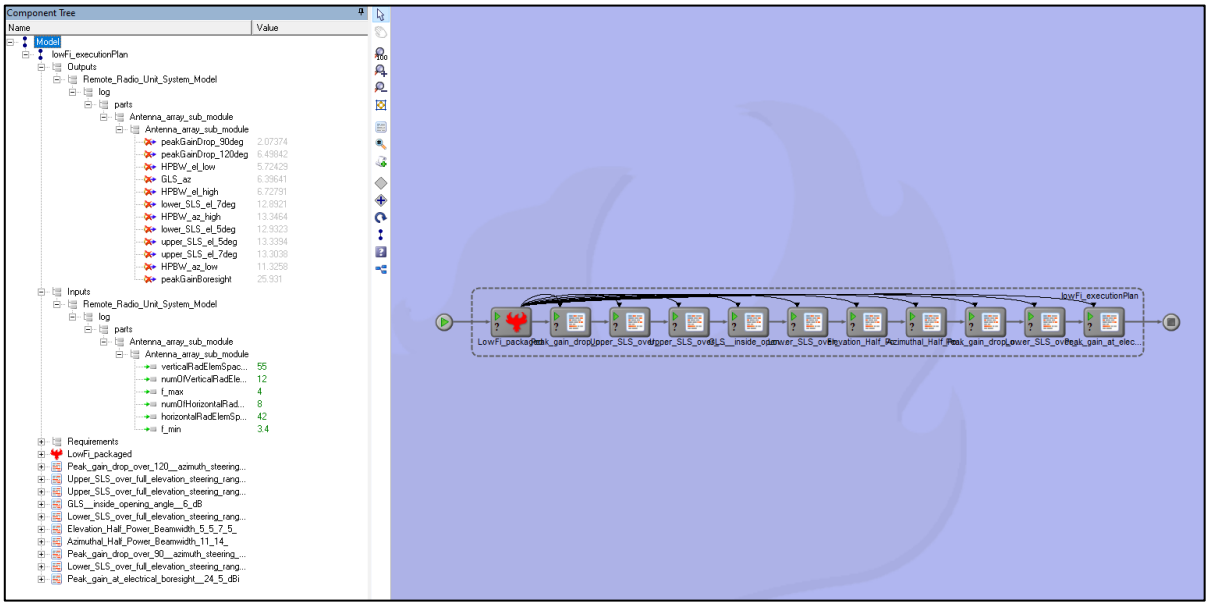

Figure 32. Antenna array analysis workflow fully integrated into a system model.

7 ANTENNA ARRAY DESIGN OF EXPERIMENTS AND OPTIMIZATION

The next and the final part of this thesis is the execution of a DOE and an optimization with a machine learning algorithm. The analysis is done with the integrated low-fidelity workflow created in the last chapter. As a consequence, the results of the DOE and optimization can only be considered to be rough approximations.

Both the DOE and optimization are performed in order to study the effects of horizontal and elevational spacings on different performance parameters of the antenna and to eventually find the best spacing option. The optimization produces a pareto front between competing performance parameter objectives of the antenna array, which is used to find the best value design alternative for the spacings of the antenna array. Finally, the resulting design and its performance parameter values are uploaded to the system model as a so-called design instance.

7.1 Antenna Array Design of Experiments

The DOE is performed using the integrated workflow created in the previous chapter, thus its execution starts with opening the DOE tool in ModelCenter when the workflow is open. First, the design variables are defined. This is done by dragging the two inputs, "verticalRadElemSpacing" and "horizontalRadElemSpacing" from the workflow's component tree to the design variables section of the DOE tool. After this, the lowest and highest values of these variables are specified. The smallest physically feasible spacing was estimated to be 45 mm vertically and 35 mm horizontally due to size of the individual radiating elements. There is no physical upper limit for spacings, however, due to the occurrence of the undesirable grating lobes at high spacings, the upper limits were set to 60 mm vertically and 50 mm horizontally. These boundaries define the design space of the DOE, or its design limits.

Next, the response variables of the DOE are defined. This is done by dragging and dropping all the outputs or performance parameters from the component tree of the workflow to the response variables section of the DOE tool. These are the parameters whose responses to the change of the spacings will be studied.

Finally, Latin-hypercube is chosen as the method or algorithm that defines the sample points of the DOE. When the number of variables is defined as X and the number of sample points as N, the algorithm can be described as working by first, dividing the ranges of all given variables into N number of equally probable intervals and then selecting one random value from each interval of each variable. The N values obtained for one variable are paired randomly with the N values obtained for another variable. This process is repeated until N X-tuples are created, which make up the Latin-hypercube sample [18]. In this example, the desired number of runs N is chosen to be 80. Figure 33 shows the resulting view in the DOE tool and highlights the design and response variables within.

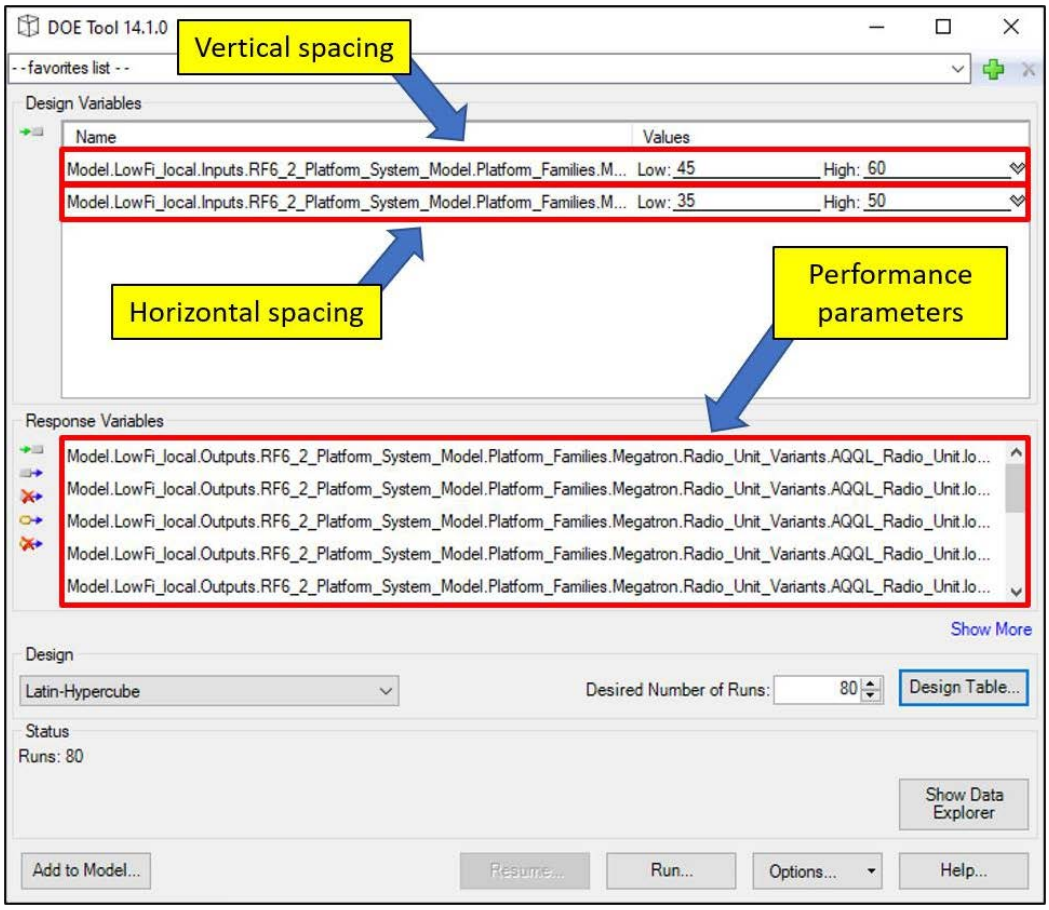

Figure 33. DOE configuration for the spacing study.

By pressing "Run" in the DOE tool, the DOE is initiated. ModelCenter executes the workflow with the 80 different design configurations created by the Latin-hypercube method. The result is a so-called trade study table that can be seen in Figure 34. The table shows the input and output values at each run of the DOE. The two first rows are the design variables, and the rest are their respective performance parameter values. The figure only shows the results of the first seven, but by scrolling, others can be brought into view.

AUTO SCROLL	1	2	3	4	5	6	7
M.L.I.R.P.M.R.A.I.p.A.A.verticalRadElemSpacing	51.6878	46.7191	52.9236	48.6514	55.0243	56.8327	52.3408
M.L.I.R.P.M.R.A.I.p.A.A.horizontalRadElemSpacing	49.6205	41.1879	39.4217	41.1276	37.7099	47.6852	38.2103
M.L.O.R.P.M.R.A.I.p.A.A.peakGainDrop_90deg	3.5306	1.52201	1.44961	1.52197	1.42776	3.92504	1.43057
M.L.O.R.P.M.R.A.I.p.A.A.peakGainDrop_120deg	11.2943	6.12512	5.34626	6.12892	5.00113	8.94383	5.11886
M.L.O.R.P.M.R.A.I.p.A.A.HPBW_el_low	6.06273	6.64739	5.9401	6.39499	5.68956	5.48637	6.00068
M.L.O.R.P.M.R.A.I.p.A.A.GLS_az	-0.49832	8.69448	13.169	8.93906	14.1618	0.43322	13.8573
M.L.O.R.P.M.R.A.I.p.A.A.HPBW_el_high	7.09627	7.94201	6.92616	7.57959	6.69111	6.49753	7.00141
M.L.O.R.P.M.R.A.I.p.A.A.lower_SLS_el_7deg	8.92613	9.77659	8.70937	9.46237	8.43033	8.08169	8.83746
M.L.O.R.P.M.R.A.I.p.A.A.HPBW_az_high	11.229	13.5488	14.1421	13.5658	14.7825	11.6857	14.5893
M.L.O.R.P.M.R.A.I.p.A.A.lower_SLS_el_5deg	11.5133	11.7341	11.433	11.6801	11.259	10.9532	11.4765
M.L.O.R.P.M.R.A.I.p.A.A.upper_SLS_el_5deg	11.5133	11.7341	11.433	11.6801	11.259	10.9532	11.4765
M.L.O.R.P.M.R.A.I.p.A.A.upper_SLS_el_7deg	8.92613	9.77659	8.70937	9.46237	8.43033	8.08169	8.83746
M.L.O.R.P.M.R.A.I.p.A.A.HPBW_az_low	9.50482	11.4855	12.0211	11.5014	12.5561	9.94628	12.3946
M.L.O.R.P.M.R.A.I.p.A.A.peakGainBoresight	26.7978	25.5384	25.9045	25.7171	25.894	26.9795	25.7299

Figure 34. The results of the DOE in a Trade study table.

Different graphs can be created from these DOE results that visualize the statistical relationships between the design variables and the performance parameters. Let us start with graphs that show the effects of horizontal radiating element spacing. One such graph can be seen in Figure 35, in which the azimuthal grating lobe suppression is shown in relation to

horizontal radiating element spacing. We can see from this figure that the azimuthal GLS decreases, i.e. the grating lobe gets stronger as the horizontal spacing is increased. At spacing values greater than 42 mm, azimuthal GLS starts to undercut its requirement according to which the azimuthal GLS must be greater than 6 dB. This gives an approximation for the maximum acceptable horizontal radiating element spacing, and shows that a higher GLS, therefore a better performance can be achieved with a small horizontal spacing.

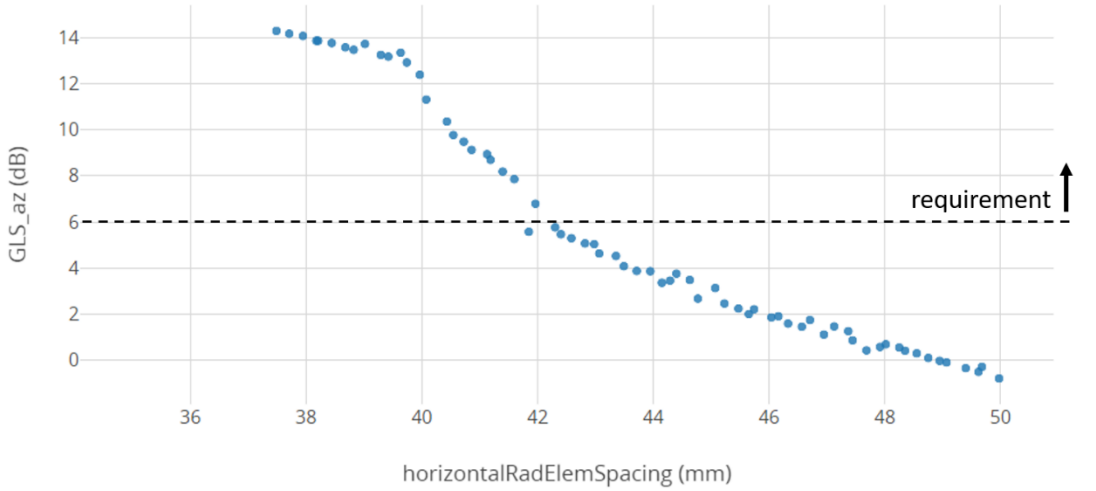

Figure 35. Azimuthal grating lobe suppression vs. horizontal radiating element spacing.

Another important relationship is revealed by comparing the horizontal radiating element spacing with peak gain drops over 90° and 120° horizontal steering ranges. This can be seen in the Figure 36. From the graph, we can see how the peak gain drop stays quite low at spacings under 41 mm, but at greater values starts to increase significantly. The drop over 90° exceeds its minimum requirement of 3 dB at a spacing of 46 mm and the drop over 120° exceed its minimum requirement of 7 dB at 43mm. Both of these values are higher than the value obtained from the GLS requirement, meaning that peak gain drop requirement has a lower restrictive effect on the spacing. However, a conclusion can be made that a lower peak gain drop, therefore a better performance, can be achieved with a small horizontal spacing.

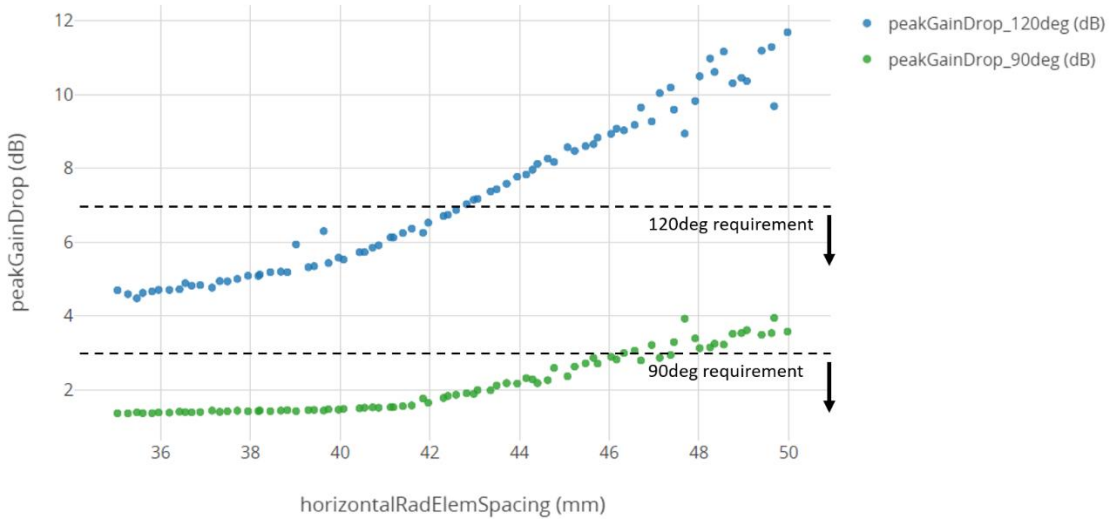

Figure 36. Peak gain drops over 90° and 120° horizontal steering ranges vs. horizontal radiating element spacing.

Yet another important result of the DOE is the relationship between the azimuth or horizontal half-power beamwidth and the horizontal radiating element spacing, which can be seen in the Figure 37. The relationship between these two can be observed to be almost linear, for both the lower 6.7th and the higher 93.3th percentile values, and that the HPBW is inversely proportional to the horizontal spacing. The higher HPBW can be seen exceeding the upper value of its requirement, 14°, at the horizontal spacing of around 40 mm, which therefore becomes an approximation for the minimum acceptable horizontal radiating element spacing. The lower 6.7th percentile HPBW can be seen falling below the lower limit of its requirement, 11°, at the horizontal spacing of around 43 mm, which is higher than the limit obtained previously with the horizontal GLS requirement.

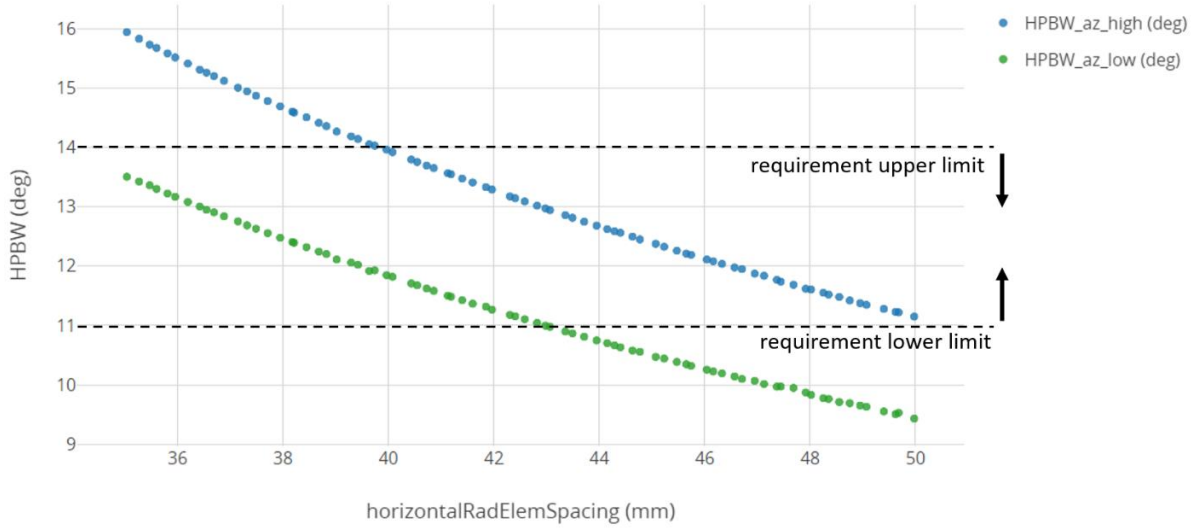

Figure 37. Azimuth half-power beamwidth vs. horizontal radiating element spacing.

The following two graphs consider the effect of vertical radiating element spacing on the performance parameters. First graph is seen in the Figure 38, which shows the relationship between the vertical radiating element spacing and the upper vertical SLS over two vertical steering ranges of 10° and 14°. An observation can be made that the relationship is close to linear for both steering ranges, and that the SLS is inversely proportional to the vertical spacing. Furthermore, the requirements for the SLS to be more than 10 dB over the 10° steering range and more than 6 dB over the 14° steering range, are both satisfied over the whole range of vertical spacing values studied with the DOE. Therefore, these requirements don't give any specific restrictions to the vertical spacing. Nevertheless, the graph shows that a higher elevation SLS, therefore better performance, is achieved with a smaller vertical spacing.

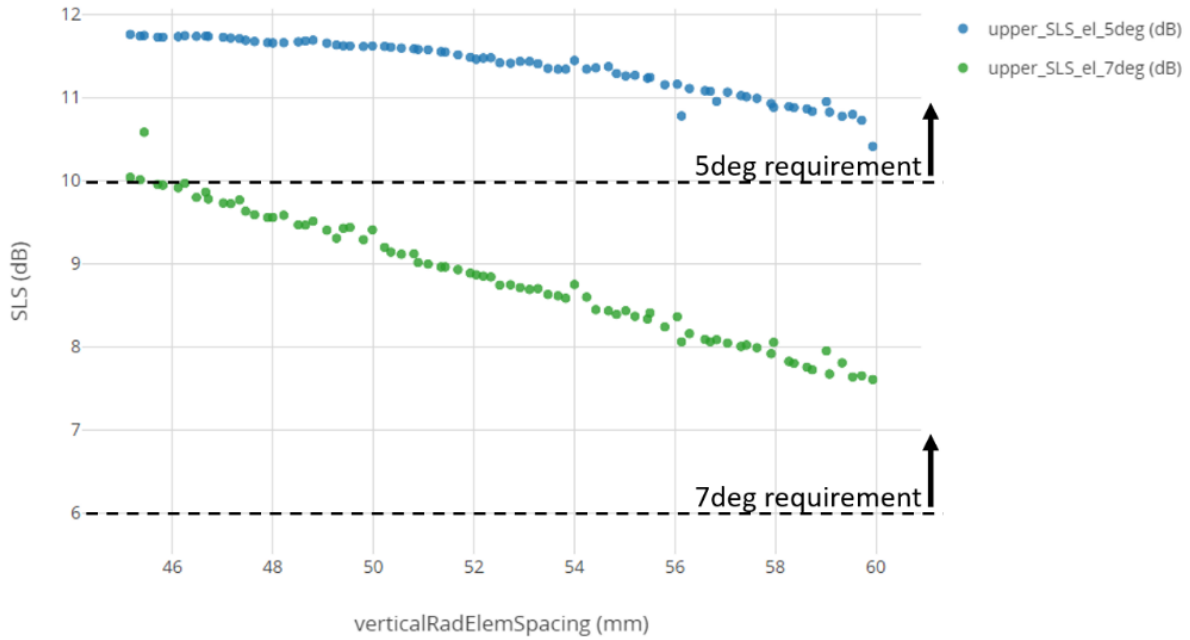

Figure 38. Upper sidelobe suppression over 14 degrees vertical steering range vs. vertical radiating element spacing.

The Figure 39 displays the relationship between the elevation or vertical half-power beamwidth and vertical radiating element spacing. The HPBW has been calculated as both, a lower 6.7th, and a higher 93.3th percentile value. An observation can be made from the graph that the vertical HPBW is inversely proportional to the vertical spacing. The higher 93.3th percentile HPBW can be seen exceeding its upper requirement of 7.5 degrees at around the vertical spacing of 49 mm, which therefore becomes an approximation for the minimum acceptable vertical radiating element spacing. The lower 6.7th percentile HPBW can be seen falling below the lower limit of the requirement, 5.5°, at vertical spacing of around 56 mm, which therefore becomes an approximation for the maximum acceptable vertical radiating element spacing.

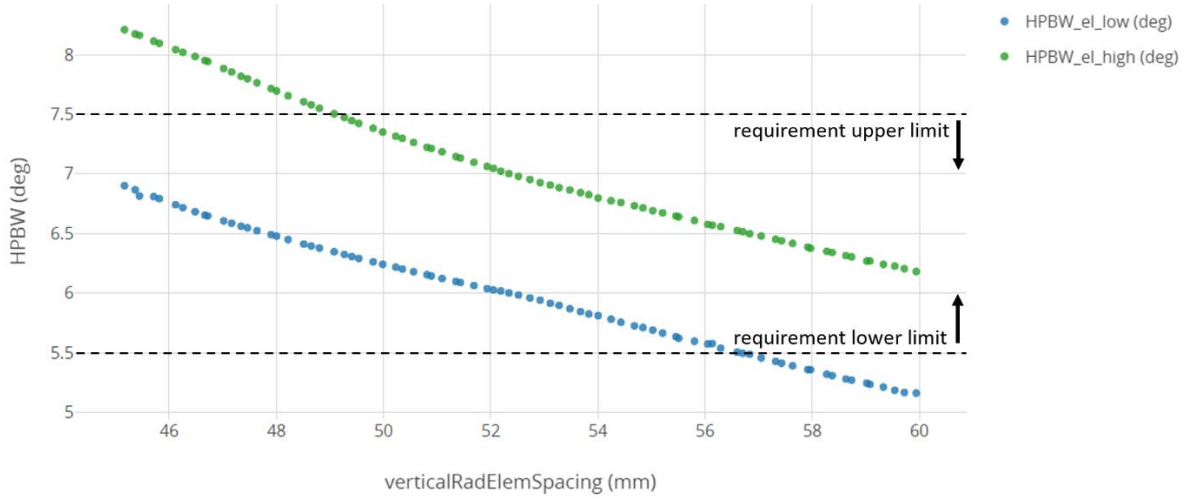

Figure 39. Elevation half-power beamwidth vs vertical radiating element spacing.

Finally, the Figure 40 is a graph that shows the peak gain at electrical boresight in relation to both horizontal and vertical spacing. An observation can be made that both spacings are directly proportional to the peak gain, thus increasing both spacings increases the peak gain. This means that in regard to gain, a better performance is achieved with high spacings.

The minimum requirement for the peak gain is 24.5 dBi, which can be seen satisfied over the whole range of both spacing values studied with the DOE. Therefore, the requirement for peak gain at electrical boresight doesn't give any specific restrictions to either spacing.

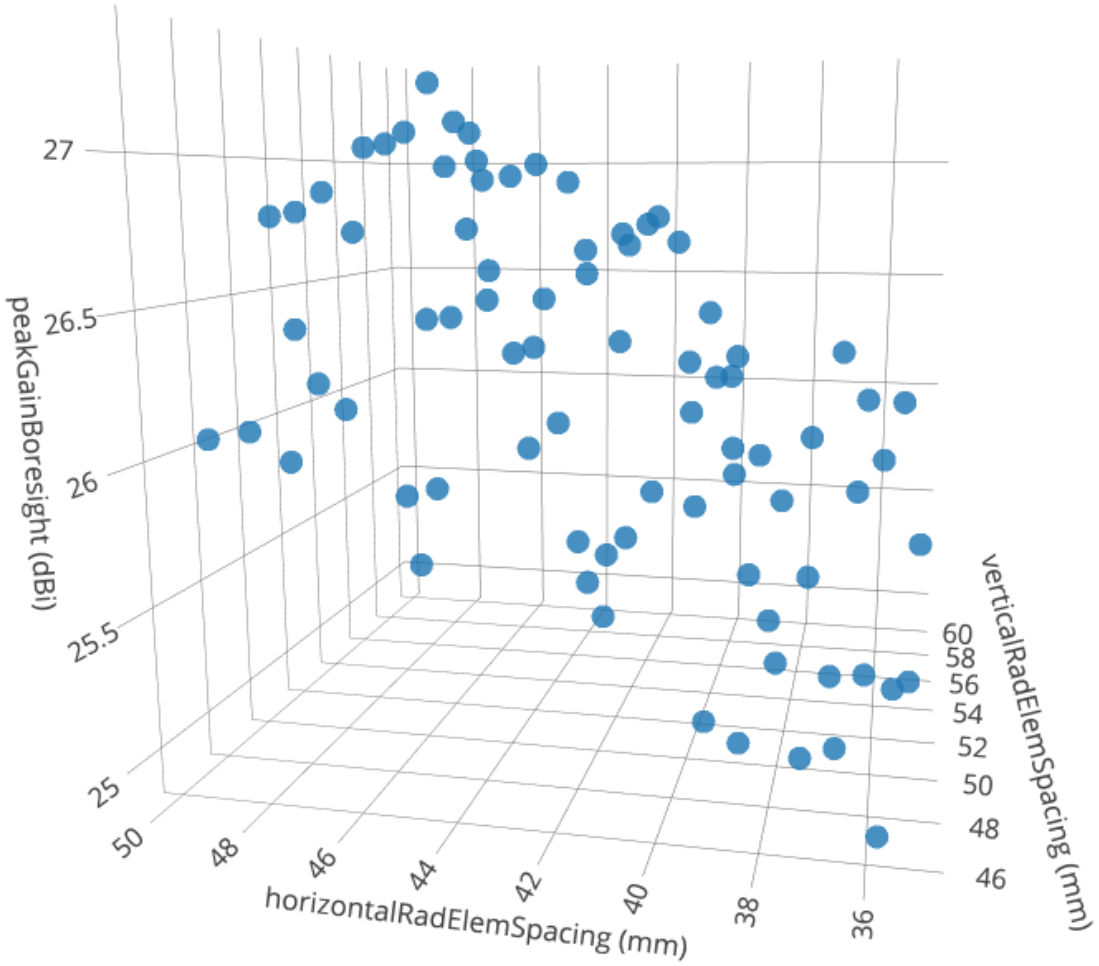

Figure 40. Peak gain at electrical boresight vs. horizontal and vertical spacing.

All these results demonstrate how powerful tool DOE can be. In this use case, it was used to determine the minimum and the maximum values for the horizontal and vertical radiating element spacings that satisfy the antenna array requirements. These values were determined to be 40mm and 42mm for the horizontal spacing, and 49mm and 56mm for the vertical spacing.

Furthermore, statistical relationships were analysed between the spacing variables and the performance parameters GLS, SLS, HPBW, peak gain drop and peak gain at electrical boresight. This provided an extensive understanding of the behaviour of the phased antenna array.

7.2 Antenna Array Optimization

Optimization starts with defining its configuration. In this use case four different objectives are defined, which are: (1) minimize the peak gain drop over 120° horizontal steering range and (2) maximize the azimuth GLS and (3) maximize vertical SLS over 14° vertical steering range and (4) maximize the peak gain at electrical boresight. These objectives are defined in the "objective definition" section of the optimization tool in ModelCenter, which can be seen in the upper part of the Figure 41.

Each requirement has a parameter called "margin" in the integrated workflow. This value is the difference between the value of requirement and the value of its corresponding performance parameter achieved with an analysis. If the margin is positive, the requirement has been satisfied by a certain amount, and on the other hand if it is negative, the requirement has not been satisfied by a certain amount. These margin values of all the requirements are added to the constraint section of the optimization tool. Therefore, by setting the lower bound of each constraint to 0, the optimizer takes the values of all the requirement as its constraints. This configuration can be seen done in the middle part of the Figure 41.

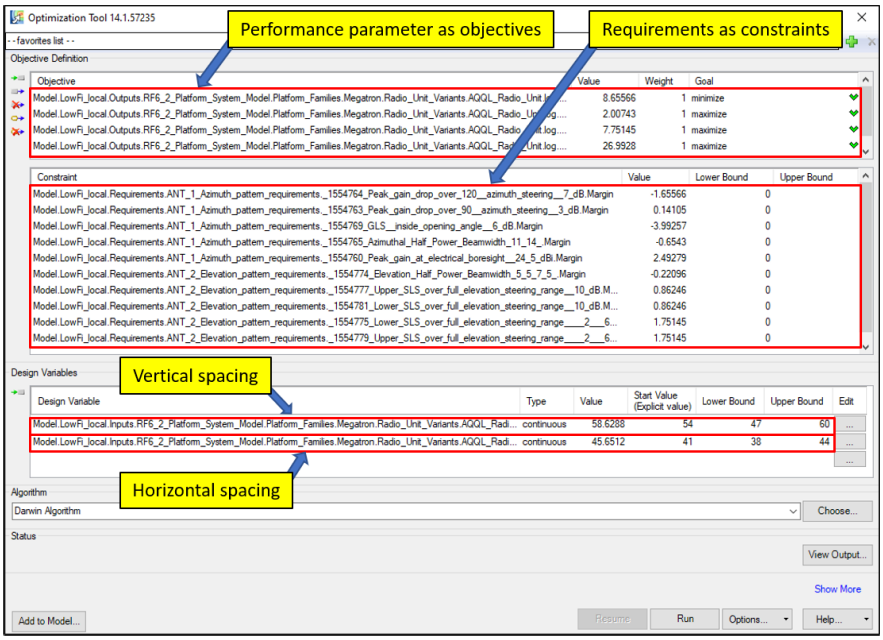

Figure 41. Optimization configuration.

Finally, the two design variables of interest, the vertical radiating element spacing and the horizontal radiating element spacing are added to the "Design Variables" section of the optimization tool. Their upper and lower bounds are set 47mm and 60mm for the vertical spacing, and 38mm and 44mm for the horizontal spacing. Even though the results of the previously performed DOE could be used here, a wider range is specified in order to demonstrate the effectiveness of machine learning driven optimization. The algorithm chosen is the previously mentioned Darwin algorithm. The algorithm's population size is set to 10, its convergence is set to happen after 5 consecutive generations without improvement, and the minimum change of pareto population required for improvement is set to 5%. After these configurations have been finished, the optimization is initiated.

The best graph to display the results of the optimization in this use case is a 3D comparison of three of its objectives. In Figure 42, we can see a comparison between the upper vertical SLS over 14° steering range, the azimuth GLS, and the peak gain at the electrical boresight. The

previously executed DOE showed that the peak gain drop is minimized and the GLS maximized by minimizing the horizontal spacing, and that the vertical SLS is maximized by minimizing the vertical spacing. On the other hand, it also demonstrated that maximizing both spacings maximizes the peak gain at electrical boresight. Thus, the first three objectives drive the optimizer to minimize both spacings, and only the last one to maximize them. For this reason, the peak gain objective gets eventually neglected by the optimizer and its convergence happens at designs with smallest possible values before the upper limit requirement for the half-power beamwidth of 14° is exceeded.

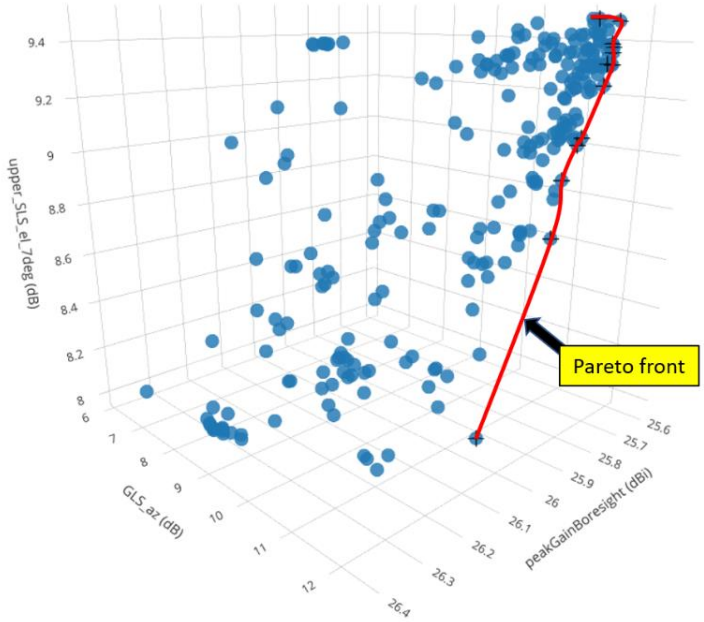

Figure 42. The optimization result with its pareto front visualized in a comparison between thee of its objective parameters.

Even with such a straightforward optimization result, a pareto front can be seen formed in the Figure 42. This is because a slightly higher GLS is caused by a higher vertical spacing. The difference between the best and the worst GLS value in the pareto front is around 1.3 dB. The difference between the best and the worst peak gain drop value in the pareto front is around 0.13 dB. The difference between the best and the worst peak gain at electrical boresight in the pareto front is around 0.55 dB. And finally, the difference between the best and the worst vertical SLS in the pareto front is around 1.25 dB. This results in a trade off between a better GLS and peak gain versus a better vertical SLS and peak gain drop.

The Figure 43 shows only the designs that are part of the pareto front. By comparing these designs to each other the best value design can be found. A design with a horizontal spacing of 39.843 mm and a vertical spacing of 50.121 mm is chosen. This design was found to offer the best value when the values of its four objective parameters are compared to each other, and all the objectives are considered equally valuable. However, another decision could be made if some performance parameter would be regarded as more valuable than others. For example, if higher gain is deemed more important than SLS, a wider vertical spacing should be chosen.

The manufacturing processes for the antenna array cannot achieve an accuracy higher than 1 mm, thus the final result for the spacing is rounded to an accuracy of 1 millimetres. The final values are 40 mm for the horizontal spacing and 50 mm for the vertical spacing.

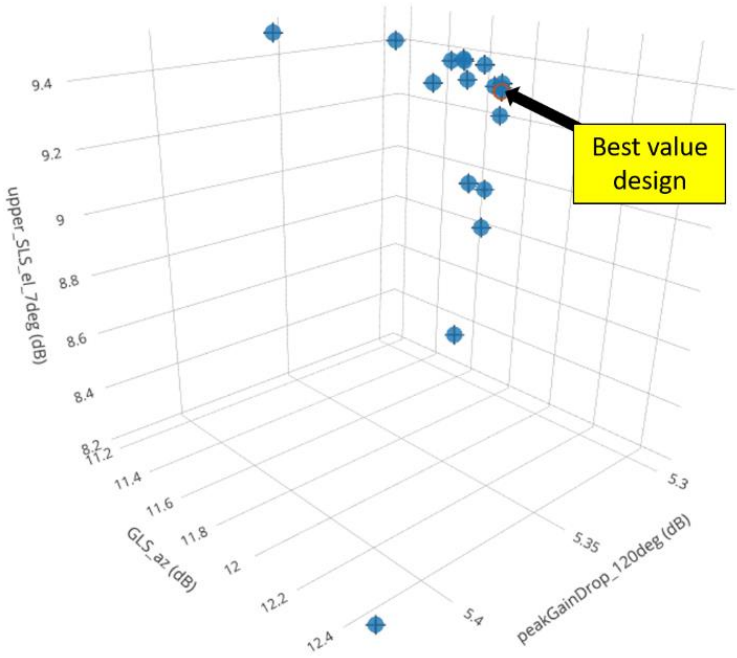

Figure 43. The best value design in the pareto front.

Now the chosen design can be uploaded to the system model. This is done in the ModelCenter MBSE plug-in. If the system model is open, the tool will automatically recognize the execution plan from which the workflow was created earlier. The tool also automatically recognizes the latest analysed design in the workflow and shows its values. By selecting "Save design instance" in the plug-in's GUI, a pop-up window appears with the system model's structure. The structure can be navigated to a location in the system model into which the design will be uploaded. In this use case it is saved to a package in the system model named "Antenna Array Designs". The resulting view in the system model can be seen in the Figure 44. This is a SysML instance element that contains the values of all the design variables and performance parameters that were connected to the system model earlier. The design information is now available in the system model, where it can be used for example in higher-level analysis.

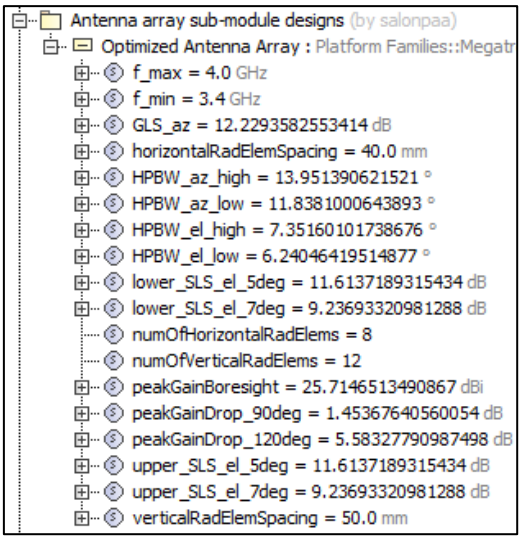

Figure 44. The design instance in the system model.

8 DISCUSSION

The first part of the thesis was a demonstration on how the modeling of hardware aspects of a phased antenna array could be done with the system modeling language SysML. The modeling of requirements with Cameo DataHub was found to create version controlling issues with DOORS Next. If a change is made to the requirement in DOORS Next, it gets automatically updated to the system model in Cameo and vice versa. One-way synchronization has to be established so that there is only one single source of truth.

The modeling of the functional architecture of the antenna array was fairly straightforward. The functionality was essentially described with two sentences. The functional architecture could be extended by including more precise beamforming functionalities, like its calibration and control that are achieved with software.

The modeling of logical architecture consisted of modeling the antenna array's logical composition and its structure. The SysML language was found an effective way to model these aspects. Two ways were used for modeling multiplicity of the internal parts of the antenna array. First one was to use the in-built multiplicity property, and the second one was to create multiple directed composition associations. The multiplicity property method resulted in a number next to the name of the part in the BDD and IBD diagrams. Multiple directed composition associations resulted in multiple copies of the part in the IBD diagram. The property method is good for modeling large multiplicities, and the association method is good for modeling small multiplicities that benefit from the actual visible copies in the model.

After this, another demonstration was given on how analysis models could be created for the antenna array with Ansys HFSS and MATLAB at three different fidelity levels and then automated in ModelCenter. The medium-fidelity model needs 16 radiation pattern plots in HFSS, but an obstacle was encountered in the automated creation of these multiple radiation patterns. For some unexplained reason, HFSS stopped responding after it had created around a dozen radiation pattern plots. This might be caused by a limited amount of random-access memory allocated for script running in HFSS. If it is an issue inside HFSS, it might be fixed in its future releases. In any case, due to this problem, both the medium and high-fidelity models could not be automated or analysed in this thesis, but nevertheless serve as exemplary ideas for implementation in the future.

The low-fidelity model was automated into an analysis workflow and then connected to the system model with ModelCenter. This demonstrated how the systems engineering and analysis domains can be integrated, which enables information from the model to be used in the analysis and analysis results to be exported to the system model.

Connecting the analysis workflow to the system model revealed some limitations of the tool. ModelCenter was found to have a poor support for multiplicity in the system model. ModelCenter considers the value properties of parts with multiplicity as arrays and doesn't supported nested multiplicity at all. This means that if for example a part has a multiplicity of 15, ModelCenter considers the part's value properties as sets of 15 different values, and value properties of parts with multiplicity within a part with multiplicity are not considered at all. This poses a challenge when ModelCenter is used with system models that contain large amounts of multiplicity of identical parts, since this requires that the value properties have only one value despite the multiplicity. Thus, it is impossible to connect value properties of parts that have multiplicity to analysis workflows in ModelCenter. Fortunately, this turned out to not be an issue in the use case presented in this thesis, since all the value properties were defined for the antenna array as a whole, i.e. the topmost element. However, if some value properties

would have been used for the inner parts of the antenna array, for example the radiating elements, they could have not been connected to the analysis workflow in ModelCenter.

After the analysis workflow had been connected to the system model, a DOE was performed on the antenna array. The DOE consisted of 80 different design configurations created by the Latin-Hypercube method for the horizontal and vertical spacings. 80 designs turned out to be a good number with this method. Both design variables were sampled with a step size of around 0.15 mm. This resulted in graphs that showed the behaviour of the performance parameters of the antenna array with its spacings. The consistency of these graphs was good, but they can only be considered as coarse approximations, since they were produced using the low-fidelity model.

Since only two design variables were used in the DOE, its results were simple and straightforward. With more design variables, the DOE can be used as powerful tool to minimize the design space before moving on to optimization. This is because it can reveal design variables that do not have a significant impact on the performance of the design, thus these variables can be ruled out from the optimization.

The results of the DOE could be used to create a metamodel for the antenna. This is a mathematical model constructed from the design points of the DOE that models the antenna array's performance continuously at different frequencies and design alternatives. Using this mathematical metamodel to analyse the behaviour of the antenna array would be much faster than simulating each design alternative and frequency with electromagnetic models in HFSS. In addition to this, the antenna array metamodel could be used together with metamodels of other components, thus analysing their combined performance and behaviour.

A machine learning optimization algorithm, called Darwin, was used to find a pareto front between the competing performance objectives of the antenna array. Three of the objectives drove both spacings to be as low as possible within limits set by the requirements. Only the objective to maximize the peak gain at electrical boresight drove the optimization to increase the spacings. Since the optimization algorithm considered all the objectives to be of equal value, the algorithm eventually converged at the smallest spacing values that satisfy the requirements.

A relationship was discovered in the optimization that was not visible in the DOE. A higher vertical spacing caused a higher horizontal GLS. This created a pareto front between the maximization of the GLS, the maximization of vertical SLS and the minimization of peak gain drop. The decision of the best design depends largely on how much importance is given to each objective. When the objectives are considered equally valuable, the design offering the best value was found to be at 40 mm of horizontal spacing and at 50 mm of vertical spacing when rounded to an accuracy of 1 mm. Since only the low-fidelity model was used to achieve this result, it can only be considered as an approximation. If all the fidelity levels could have been used, the design space would have been gradually narrowed down with the low and medium-fidelity models and the final design decision would have been done based on the results achieved with the high-fidelity model.

Finally, the best value design was uploaded to the system model. This is an important step in sharing the design information between the systems engineering and analysis domains. System engineers working with the system model could use this design information to create specifications for the antenna array. If multiple design are uploaded, they could be compared with each other and analysed together with other parts or components of the system.

ModelCenter includes additional tools that could be used to perform robustness analysis for the antenna array. The purpose of this type of analysis is to find designs that are insensitive to manufacturing tolerances. This is done by running Monte-Carlo analysis on the workflow; however, an accurate Monte-Carlo analysis might require thousands of workflow executions.

A lot of time could be saved by first, creating a metamodel, for instance a response surface model from the workflow with ModelCenter's algorithms, and then running the Monte-Carlo analysis on it instead of the actual workflow. If the response surface model used is accurate, the analysis can provide graphs that show the robustness of each design variable when compared with all the performance parameters. Furthermore, robustness analysis could be combined with optimization so that both the robustness and the performance of the design are optimized.

9 SUMMARY

This thesis introduced the concept of Model-based systems engineering and provided an example on how the hardware aspects of a phased antenna array can be modeled with a system modeling language SysML in a system modeling software Cameo Systems Modeler, and demonstrated how the resulting system model can be used as central hub for integration with the analysis of a phased antenna array.

This thesis covered the creation of analysis models at three different fidelity levels for a phased antenna array that operates in the frequency range from 3.4 to 4 gigahertz. The models were created with the electromagnetic modeling and simulation software Ansys HFSS, and the programming and numeric computing platform MATLAB, which was used to create a script that handled post-processing of the simulation results.

The lowest fidelity model was automated in the Multi-disciplinary analysis and optimization tool ModelCenter. The result was an analysis workflow with configurable design parameters as its inputs and performance evaluation parameters as its outputs. The workflow combined and automated the consequent execution of the HFSS electromagnetic model and the post-processing MATLAB script. Afterwards, the workflow was integrated with the system model, which enabled the use of requirements in the analysis, and the ability to upload designs achieved with the analysis to the system model.

This connected workflow was used to perform a design of experiments and a machine learning driven optimization on the phased antenna array, with the goal of finding the best possible spacings between the individual radiating elements in the array. The design of experiments produced graphs that visualized statistical relationships between the design parameters of the antenna array and its performance evaluation parameters.

The optimization produced a graph that visualized a pareto front between different performance evaluation parameters of the antenna array. In other words, the graph showed the design alternatives that cannot be further improved in any parameter without degrading another.

This graph was used to make an informed decision on the best radiating elements spacings in the antenna array and the design was uploaded to the system model. This demonstrated system and analysis modeling and their integrated usage in the design and optimization of a phased antenna array.

10 REFERENCES

[1] INCOSE. (read 09.06.2021) General System Definition. URL: https://www.incose.org/about-systems-engineering/system-and-se-definition/general-system-definition

[2] 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. (2000).

[3] Fernández Pérez, J. L., & Hernandez, C. (2019). Practical Model-Based Systems Engineering. Artech House.

[4] INCOSE. (read 09.06.2021) Systems Engineering Definition. URL: https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition

[5] INCOSE. (read 02.06.2021) Engineered System Definition. URL: https://www.incose.org/about-systems-engineering/system-and-se-definition/engineered-system-definition

[6] Systems Engineering Vision 2020. (2007). INCOSE

[7] Sanford Friedenthal, Alan Moore, & Rick Steiner. (2015). A Practical Guide to SysML: The Systems Modeling Language: Vol. Third edition. Morgan Kaufmann.

[8] M. Schluse, L. Atorf and J. Rossmann. (2017) Experimentable digital twins for model-based systems engineering and simulation-based development. In: Annual IEEE International Systems Conference (SysCon), April 24-27, Montreal, QC, Canada, pp. 1-8.

[9] D. Kaslow, G. Soremekun, H. Kim and S. Spangelo. (2014) Integrated model-based systems engineering (MBSE) applied to the Simulation of a CubeSat mission. In: IEEE Aerospace Conference, March 1-8, Big Sky, MT, USA, pp. 1-14.

[10] OMG. (read 09.06.2021) What is SysML. URL: https://www.omgsysml.org/what-is-sysml.htm

[11] Balanis, C. A. (2016). Antenna theory : Analysis and design. John Wiley & Sons, Incorporated.

[12] 145-2013 – IEEE Standard for Definitions of Terms for Antennas. (2014).

[13] NGMN. Recommendation on Base Station Antenna Standards. (V11.1) (2019)

[14] Lau, H. (2019). Practical Antenna Design for Wireless Products. Artech House.

[15] Y. Jin and B. Sendhoff. (2009) A systems approach to evolutionary multiobjective structural optimization and beyond. IEEE Computational Intelligence Magazine, Vol. 4, no. 3, pp. 62-76.

[16] M. Poian, S. Poles, F. Bernasconi, E. Leroux, W. Steffe and M. Zolesi. (2008) Multi-objective optimization for antenna design. In: IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems, May 13-14, Tel-Aviv, Israel, pp. 1-9.

[17] Phoenix Integration & Northrop Grumman. (read 17.08.2021) A Model-Based Systems Engineering (MBSE) Approach to the Design and Optimization of Phased Array Antenna Systems Webinar. URL: https://vimeo.com/551281413.

[18] J.C. Helton, F.J. Davis. (2003) Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. In: Reliability Engineering & System Safety, Volume 81, Issue 1, pp. 23-69.